# Privacy-Preserving Data Publishing

**Masooma Iftikhar**

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

September 2022

# Certificate Of Authorship/Originality

I, Masooma Iftikhar, declare that this thesis is submitted in fulfilment of the requirements for the degree of Doctor of Philosophy, in the School of Computing at the Australian National University.

I certify that this thesis is my own original work, except where otherwise indicated. I also certify that this document has not been submitted for obtaining an award at any other academic institution.

Masooma Iftikhar
22 September 2022

Dedicated to my dear parents and my dear nephew Hadi Ali.

# Acknowledgements

First and foremost, I would like to thank Allah for the wisdom and perseverance that He has been bestowed upon me during this research work, and indeed, throughout my life.

It is an immense pleasure to thank the many people who made this thesis possible.

I would like to express my sincere gratitude to my supervisors, Dr. Qing Wang, Dr. Yu Lin, and Dr. Ramesh Sankaranarayana for their continuous guidance and encouragement over the past few years. I owe my supervisors a debt of appreciation for sharing their wisdom and showing me new ways of doing research.

I would like to extend my special gratitude to Dr. Qing Wang for her continuous support throughout my research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and writing of this dissertation. I could not have imagined having a better mentor for my Ph.D. degree. Throughout my research, she provided encouragement, sound advice, constructive feedback, support, and helped me towards becoming an independent researcher.

Besides my supervisors, I would like to thank my collaborator, Dr. Yang (Kelvin) Li, for his support and constructive feedback. Moreover, I am indebted to my friends, Humaira, Sonia, Rafia, and also many colleagues and friends at The Australian National University and Graduate House for their extensive support, quality time, and understanding. I also would like to acknowledge all kinds of help and support provided by administrative staff, the HDR Team, and many others at the CECS ANU.

Last but not least; I would like to thank my entire family for providing encouragement and motivation. I am grateful for having great parents, Sheikh Iftikhar Ali Zafri and Naheeda Zafri, who planted the seeds of greatness in me and cultivated my values. I cherish the extensive support that I received from my family members, Iqtadar Ali, Waqar Ali, Saba Batool, and Tathir Zahra. I am especially grateful to my niece and nephews, Hadi Ali, Zoha Ali, and Mohib Ali, who dedicated their endless love to me. Most of all, my exceptional gratitude goes out to my loving husband, Ali Raza, for his unconditional emotional support and patience throughout my degree. He selflessly accompanied me during the ups and downs of academic research and encouraged me throughout the obstacles and challenges of my studies. It would have been impossible to find the inspiration and motivation needed to deliver this work without the help of such an incredible man.

# Publications

This thesis is based on the following contributions which have been published across multiple peer-reviewed journals and conferences. A list of publications in reverse chronological order is given below:

[1] **Iftikhar, M.**, Wang, Q., Li, Y. "dK-Personalization: Publishing Network Statistics with Personalized Differential Privacy" *The 26th Pacific-Asia Conference On Knowledge Discovery And Data Mining (PAKDD)*, 2022. (to appear)

The concepts, formulation, development of the dK-Personalization framework and proposed approaches, sensitivity analysis, implementation and evaluations of this paper are presented in Chapter 8.

[2] **Iftikhar, M.**, Wang, Q. "dK-Projection: Publishing Graph Joint Degree Distribution with Node Differential Privacy" *The 25th Pacific-Asia Conference On Knowledge Discovery And Data Mining (PAKDD)*, 2021.

The concepts, formulation, development of the dK-Projection framework and algorithm, sensitivity analysis, algorithm implementation and evaluations of this paper are presented in Chapter 7.

[3] **Iftikhar, M.**, Wang, Q., Lin, Y. "dK-Microaggregation: Anonymizing Graphs with Differential Privacy Guarantee" *The 24th Pacific-Asia Conference On Knowledge Discovery And Data Mining (PAKDD)*, 2020.

The concepts, formulation, development of the dK-Microaggregation framework and algorithms, theoretical analysis, complexity analysis, algorithm implementation and evaluations of this paper are presented in Chapter 6.

[4] **Iftikhar, M.**, Wang, Q., Lin, Y. "Publishing Differentially Private Datasets via Stable Microaggregation" *The 22nd International Conference on Extending Database Technology (EDBT)*, 2019.

The development of the new framework for publishing differentially private datasets, theoretical discussion, algorithm implementation and evaluations of this paper are presented in Chapter 4.

[5] **Iftikhar, M.**, Wang, Q., Li, Y., Lin, Y., Shao, J. "Differentially Private Data Release via $\alpha$-Stable Microaggregation" *Data Mining and Knowledge Discovery*. (under journal preparation).

The concepts, formulation, development of the $\alpha$-Stable Microaggregation framework and algorithms, order property, theoretical analysis, complexity analysis, algorithms implementation and evaluations of this paper are presented in 5.

# Contents

# List of Figures

# List of Tables

# Abstract

With the advances of data analytics, preserving privacy in publishing data about individuals becomes an important task. The data publishing process includes two phases: (i) data collection phase, and (ii) data publishing phase. In the data collection phase companies, organizations, and government agencies collect data from individuals through different means (such as surveys, polls, and questionnaires). Subsequently, in the data publishing phase, the data publisher or data holder publishes the collected data and information for analysis and research purposes which are later used to inform policy decision making. Given the private nature of collected data about individuals, releasing such data may raise privacy concerns, and there has been much interest to devise privacy-preserving mechanisms for data analysis. Moreover, preserving privacy of an individual while enhancing utility of published data is one of the most challenging problems in data privacy, requiring well-designed privacy-preserving mechanisms for data publishing.

In recent years, differential privacy has emerged as one formal notion of privacy. To publish data under the guarantees of differential privacy, there is a need for preserving data utility, along with data privacy. However, the utility of published data under differential privacy is often limited, due to the amount of noise needed to achieve differential privacy. One of the key challenges in differentially private data publishing mechanisms is to simultaneously preserve data privacy while enhancing data utility. This thesis undertakes this challenge and introduces novel privacy-preserving mechanisms under the privacy guarantee of differential privacy to publish individuals' data while enhancing published data utility for different data structures.

In this thesis, I explore both relational data publishing and graph data publishing. The first part of this thesis will consider the problem of generating differentially private datasets by integrating microaggregation into the relational data publishing methods in order to enhance published data utility. I formulate a general framework to characterize a desired property of microaggregation and further develop simple yet effective algorithms for publishing relational data under differential privacy. The proposed framework outperforms the state-of-the-art methods by providing better within cluster homogeneity and also reducing noise added into differentially private datasets significantly.

The second part of this thesis will consider graph data publishing. Since graph data is highly sensitive to structural changes. Perturbing graph data for achieving differential privacy inevitably leads to injecting a large amount of noise and the utility of graph data is severely limited. Moreover, when applying differential privacy to network data, two interpretations of differential privacy exist: *edge differential pri-*

*vacy* (edge-DP) and *node differential privacy* (node-DP). Under edge-DP, I propose a microaggregation-based framework for graph anonymization which preserves the topological structures of an original graph at different levels of granularity through adding controlled perturbation to its edges. Within the proposed framework, I further develop a simple yet effective microaggregation algorithm under a distance constraint. The proposed framework can significantly reduce noise added to achieve differential privacy over graph data, and thus enhance the utility of anonymized graphs. Under node-DP, I study the problem of publishing higher-order network statistics. This problem is known to be challenging since even simple graph statistics (e.g., edge count) can be highly sensitive to adding or removing a single node in a graph. To address this challenge, I propose a general framework for publishing graph statistics and develop a novel graph projection algorithm to transform graphs for controlled sensitivity. Furthermore, I consider personalization to achieve personal data protection under personalized (edge or node) differential privacy while enhancing network data utility. To this extent, four approaches are proposed to handle the personal privacy requirements of individuals.

I have conducted extensive experiments using real-world datasets to verify the utility enhancement and privacy guarantee of the proposed frameworks against existing state-of-the-art methods to publish relational and graph data.

Chapter 1

# Introduction

Data related to individuals collected by the governments, corporations, and organizations has produced remarkable opportunities for information-based decision making and helped researchers to perform statistical analysis on individuals. Similarly, the benefits of prompt collection and sharing of data are undeniable as highlighted by the recent outbreak of the COVID-19 pandemic where timely collection and sharing of public health data are needed to provide information for policy makers and frontline workers to make rapid and informed decisions [31]. However, data in its original form contains sensitive or personally identifiable information, such as medical history, private communication records, and relationships among individuals. Therefore, it is important that sharing of such data is balanced with protecting the confidentiality of individuals.

## 1.1 Data Publishing Process

The data publishing process includes data holders/publishers who collect data from individuals (e.g., health data or census data) and later release the collected data for the use of data recipients to perform analysis and research. Generally, such data is stored in tables where each row (a record) corresponds to a distinct individual or stored in networks where each individual corresponds to a distinct node in a network. Entities involved in the data publishing process are shown in Figure 1.1.

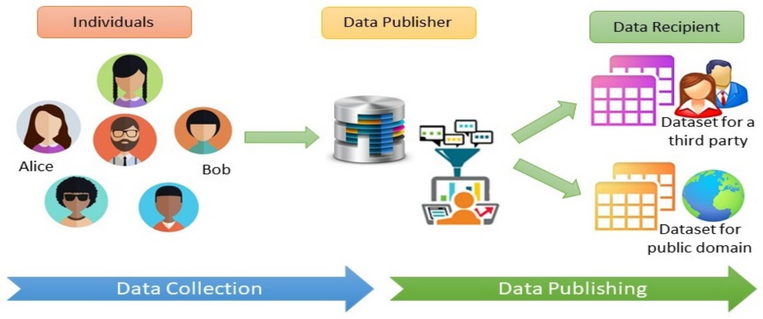


**Figure 1.1**: Entities involved in the data publishing process.

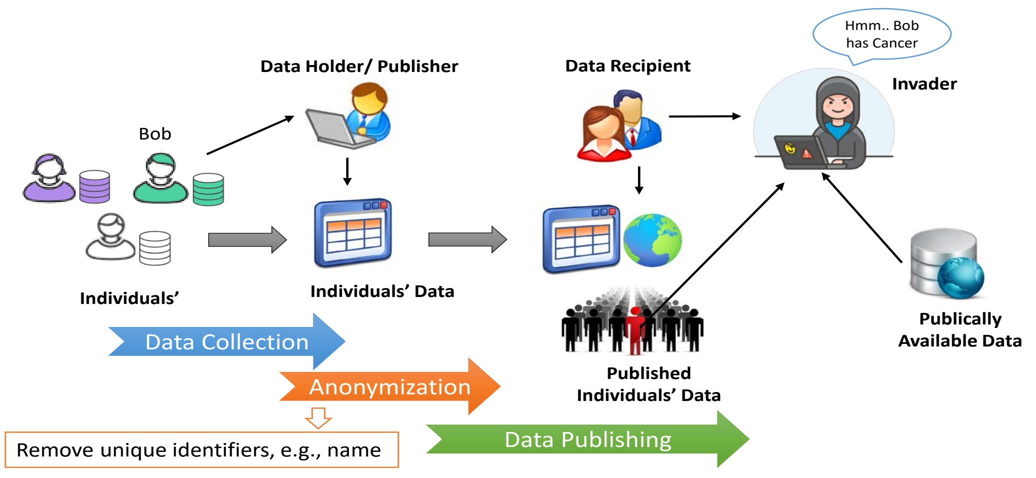


**Figure 1.2**: An overview of data publishing process.

The objective of the data publisher is to publish data in such a way that allows useful analysis while protecting privacy of an individual. Since data analysis can be performed in the absence of individual identifiers (e.g., name or social security number), the data publisher first removes identifying attributes. However, such anonymization may be insufficient because even when data is anonymized without publishing any identity information, an individual may still be revealed based on publically available data [92; 26]. One assumption in privacy-preserving data publishing (PPDP) is that the data recipient could be an invader who has some background knowledge or information. The background knowledge is described by a set of variables which can be the attributes of a released table/graph. Hence, an invader can use available information to reveal an individual's identity and associated sensitive data. As depicted in Figure 1.2, the data publisher collects data from individuals and performs data sanitization by anonymizing unique identifiers such as names, before publishing data for the use of data recipients. An invader can use other publically available data (e.g., census data) to link an individual to a particular record/node and infer sensitive information of an individual. Thus, there is a pressing need to address privacy concerns in the data publishing process.

In general, there are two types of information disclosures being identified in literature [24; 60; 63], i.e., *identity disclosure*, and *attribute disclosure*. Once there is identity disclosure occurring, an individual is re-identified and the corresponding sensitive values are revealed. Identity disclosure often leads to attribute disclosure; however, attribute disclosure can occur with or without identity disclosure [63]. Attribute disclosure occurs when new information about some individuals is revealed and the invader is able to determine some new characteristics of an individual based on released data. Moreover, an invader has different types of information, e.g., the actual dataset that has been published, some information about individuals in this published dataset (e.g., zip code), and other publicly available data (e.g., census data). This information can be used by the invader to reveal sensitive information of an individual, thereby requiring privacy-preserving mechanisms to publish data.

## 1.2 Privacy-Preserving Data Publishing

With the advances of data analytics, preserving privacy in sharing individuals' data becomes an important task. As a result, privacy-preserving data publishing (PPDP) has received significant considerations in research communities. One of the key challenges in PPDP is to simultaneously preserve data privacy and data utility (information usefulness) in anonymous data. On the other hand, a privacy mechanism must satisfy composition or allow graceful degradation of privacy with multiple invocations on the same data [17; 35]. Furthermore, the output of a privacy mechanism must not change the privacy guarantee [54; 69]. Thus, various anonymization techniques [92; 67; 63; 102; 100; 113; 25] have been proposed for different data publishing circumstances. Among early works, $k$-anonymity [96] is a privacy model widely applied to guarantee data privacy of individuals. The popularity of $k$-anonymity has led to various attempts to address the limitations of $k$-anonymity [96]. On the other hand, *differential privacy* (DP) [29; 25] – *the focus of this thesis* – is a robust privacy notion that allows statistical analysis of sensitive data while providing strong privacy guarantees.

Differential privacy is a mathematical definition for privacy loss when the private information of an individual is released. Differential privacy ensures that the output of a computation undergoes enough perturbation to mask whether an individual is present or not in the output. Thus, an invader cannot infer the presence or absence of an individual in the input based on any output. The magnitude of random noise for perturbation is determined by the sensitivity of a query function (i.e., the maximum impact that one individual can have on the output) and a global privacy parameter $\varepsilon \in [0, \infty)$, also called privacy level/budget, where a smaller value of $\varepsilon$ implies a stronger privacy guarantee and requires larger noise.

With the emergence of differential privacy [29; 25] as a widely recognized mathematical framework for privacy, a number of works have considered relational data settings to release differentially private datasets [68; 105]. On the other hand, a line of works [40; 82] have explored network data settings and have investigated the problem of publishing anonymized graphs under the guarantee of differential privacy. However, when generating differentially private data, there is always a trade-off being made between privacy and utility in published data because privacy is attained at the cost of accuracy by adding noise to the data [44]. In this thesis, I explore both areas of research, to tackle this challenge, and develop privacy-preserving mechanisms under the framework of differential privacy to publish relational data and graph data while enhancing output utility.

### 1.2.1 Relational Data Publishing

In recent years, a number of works considered releasing differentially private datasets [68; 105]. Such differentially private datasets can guarantee differential privacy controlled by a privacy parameter $\varepsilon$ in a robust statistical way. Broadly speaking, there are two types of approaches to release differentially private data in the literature: one is based on histograms [104; 106; 103], and the other is based on record perturbation [105; 21]. There are some limitations in the histogram-based approaches including an exponential growth of the number of histogram bins with the number of attributes and being limited to histogram queries [12; 89; 91]. Conversely, approaches based on record perturbation entail a large amount of noise being added into the results of queries [105; 91]. Nevertheless, these approaches are not limited to histogram queries and allow dealing with multiple attributes.

The utility of differentially private datasets is limited because of noise being added to guarantee differential privacy. Preferably, there is a need to preserve the privacy of individuals but still preserve the utility for executing statistical analysis on individuals data. One of the record perturbation approaches [21] used microaggregation to achieve $k$-anonymity. Later, in [91] a microaggregation-based mechanism, i.e., *insensitive microaggregation*, has been proposed. In this work, $\varepsilon$-differentially private datasets are generated via record masking, in which microaggregation is adapted as a middle step to reduce the sensitivity of differentially private datasets. It uses microaggregation to attain $k$-anonymity in which a particular correspondence among clusters in the microaggregated datasets of two neighboring datasets is enforced. In doing so, the amount of noise added to differentially private datasets can be greatly reduced and the utility gets enhanced. However, the existing works, including MDAV [21] and insensitive microaggregation [91], either produce a low degree of within cluster homogeneity or fail to reduce the amount of noise independent of the size of a dataset [89]. To alleviate these limitations, the first part of this thesis will consider the problem of generating $\varepsilon$-differentially private datasets by integrating microaggregation into relational data publishing in order to enhance published data utility.

### 1.2.2 Graph Data Publishing

Graph data analysis has been widely performed in real-life applications. For instance, online social networks are explored to analyze human social relationships, election networks are studied to discover different opinions in a community, and co-author networks are used to understand collaboration relationships among researchers [101]. Additionally, graph analytics can provide unique and rich insights about social network activities, disease transmission, consumer behaviour, communication patterns, and recommendations [85]. However, such networks often contain sensitive or personally identifiable information, such as social contacts, personal opinions and private communication records. Given the private nature of data about individuals stored in networks, releasing graph data raises privacy concerns. There has been much interest to devise privacy preserving mechanisms for graph data analysis [50; 14]. To preserve graph data privacy, various anonymization techniques for graph data publishing have been proposed in the literature [10; 41; 66; 115]. Nonetheless, even when a graph is anonymized without publishing any identity information, an individual may still be revealed based on structural information of a graph [41].

Differential privacy (DP) [29] on graphs has received increasing attention, since it offers a robust privacy guarantee while making no assumptions about the prior knowledge of an adversary. When applying differential privacy to network data, two variants of differential privacy were introduced [40]: 1) *edge differential privacy* (edge-DP), which aims to hide changes on a single relationship, i.e., the addition or deletion of an edge in a network, 2) *node differential privacy* (node-DP) which aims to hide changes of an individual and all of the relationships associated with such an individual, i.e., the addition and deletion of a node and all of its incident edges in a network. However, it has been acknowledged in the literature [14; 53; 95] that node differential privacy is more challenging to achieve than edge differential privacy, since deleting one node may cause, in the worst case, the deletion of $|V| - 1$ edges, where $V$ is the set of all nodes in a network.

Early works [40; 88; 85; 101; 97; 11; 50] on differentially private network data focused on *edge-DP*. Although these techniques are promising, they can only achieve $\varepsilon$-differential privacy over a graph by injecting the magnitude of random noise proportional to the sensitivity of queries, which is fixed to global sensitivity. Due to the high sensitivity of graph data on structural changes, the utility of anonymized graphs being published by these works is rather limited. Different from existing works, my aim in this thesis is to anonymize graphs under edge-DP using less sensitive queries which can reduce the overall noise needed to achieve $\varepsilon$-differentially private graphs.

Some recent studies [53; 16; 84; 14] have attempted to tackle the challenge of *node-DP*. However, these studies are limited to publishing only simple network statistics (e.g., edge count, triangle count, and degree distribution) in order to maintain relatively low sensitivity under node-DP. Different from these existing works, I aim to release higher-order network statistics by effectively controlling sensitivity under node-DP.

One fundamental shortcoming of differential privacy is that, a uniform privacy level (i.e., $\varepsilon$) is assigned to each individual while performing perturbation; however, in practice, different individuals may have different privacy levels based on their own preferences subject to their data [49; 32]. For instance, in social networks an individual (user) tends to share their personal information with their close ones and only shares obscured data with acquaintances or strangers. Therefore, differential privacy may lead to providing insufficient protection for some individuals, while over-protecting others. Different from existing works, I aim to release network data distributions under edge-DP [50] and node-DP [14] with personalization while enhancing network data utility.

The second part of this thesis will consider graph data publishing and present novel PPDP mechanisms for publishing higher-order graph statistics under edge, node and personalized differential privacy while enhancing published graph data utility.

## 1.3 Research Objectives and Challenges

The goal of this thesis is to develop privacy-preserving data publishing mechanisms under the framework of differential privacy for different data structures. I aim to explore relational data and graph data with the following research questions:

- How to determine the right amount of noise that preserves privacy while enhancing output utility under differential privacy for different data structures?
- How to reduce the amount of noise needed to achieve differential privacy under different data structures?
- How to enhance data utility under different data structures while providing differential privacy guarantees?

Developing privacy-preserving data publishing mechanisms under differential privacy for different data structures brings up several challenges. For relational data, when a dataset is large and contains multiple attributes, the overall noise needed to achieve differential privacy is high. Thus the output utility is decreased, accordingly. On the other hand, graph data is highly sensitive to structural changes. Directly perturbing graph data often leads to injecting a large amount of random noise and the output utility can be severely impacted. Moreover, preserving topological structures of an original graph while achieving differential privacy is not straightforward. This requires a careful analysis on the sensitivity of queries and accordingly to design a way of controlling or reducing sensitivity. Generally, adding more noise provides better privacy but less utility, and vice versa.

The contributions of this thesis are guided by the following research objectives:

**Objective I -** The first research objective of this thesis is to design and develop a privacy-preserving data publishing framework for relational data. The aim is to publish differentially private datasets for relational data which meet the following requirements:

- *Privacy:* Provide the privacy guarantees of differential privacy.
- *Utility:* Enhance the accuracy of published data while providing differential privacy.

To increase the overall utility, the key challenge is how to enhance utility of published data by reducing the magnitude of random noise proportional to the sensitivity, in comparison with the state-of-the-art methods.

**Objective II -** The second research objective of this thesis is to seek solutions to share meaningful information about networks while preserving privacy under differentially privacy. In this regard, I first aim to apply differential privacy to graph data by considering edge-DP, under the rationale that edge-DP protects relationship between two individuals (nodes in a network) from being disclosed. The challenge of anonymizing graphs under edge-DP is that any mechanism for sharing graphs must deal with the tension between two goals: protecting privacy and achieving structural similarity to an original graph. A desired solution must meet the following requirements:

- Differential privacy is guaranteed for protecting relationship between individuals in an original graph through adding controlled perturbation to its edges.
- The topological structures of an original graph can be preserved as much as possible at different levels of granularity.
- The utility of an anonymized graph is enhanced by reducing the magnitude of noise needed to achieve edge-DP.

Graph data is highly sensitive to structural changes. Perturbing graph data for achieving differential privacy inevitably leads to injecting a large amount of noise and the utility of anonymized graphs is severely limited. To deal with this issue, several works [85; 97; 98; 101] have explored techniques of indirectly perturbing graph data through graph abstraction models. The central ideas behind these studies are to first project a graph into a statistical representation, and then add random noise to perturb such representations. Although these techniques can achieve edge-DP but inject random noise proportional to the sensitivity of queries, which is fixed to global sensitivity. Thus, utility of anonymized graphs being published by these works is limited because of high global sensitivity. My aim is to anonymize graphs under edge-DP using less sensitive queries to reduce the amount of noise added into differentially private graphs for enhancing utility of anonymized graphs.

**Objective III -** The third research objective of this thesis is to investigate the problem of publishing higher-order graph statistics and develop a simple yet effective solution that can approximate graph statistics as much as possible while satisfying node-DP. When applying node-DP to graph data, the purpose is to approximate the effect of providing control of personal data to individuals. To justify edge-DP, the same argument cannot be applied, as in graph data, a node and its related edges represent personal data of an individual. However, an edge does not necessarily represent data controlled by one individual. An invader may attack graph data with an aim of re-identifying nodes, which illustrates that the privacy concern may also exist at individual's level, i.e., node level. In general, the privacy offered by node-DP is much stronger than edge-DP, and it is acknowledged that node-DP is more challenging to design and implement [5; 14; 95; 53]. I aim to explore the following research questions:

- Is it possible to design a mechanism to publish higher-order graph statistics under node-DP, which can efficiently enhance output utility?
- How to determine the sensitivity to publish higher-order graph statistics that is needed to achieve node-DP?

- How to reduce the amount of noise needed to achieve node-DP by controlling sensitivity effectively?

These questions are challenging, since even simple graph statistics (e.g., edge count) can be highly sensitive to adding or removing a single node in a graph. On the other hand, the existing studies [53; 84; 14; 16] have only studied the release of simple statistical data of networks, i.e., edge count [53; 95], counts of small subgraphs [5; 16], and degree distribution [53; 84; 14] in order to maintain relatively low sensitivity under node-DP. Different from existing work, I aim to release higher-order graph statistics while enhancing output utility under node-DP.

**Objective IV -** The fourth research objective of this thesis is to offer personalization so as to achieve personal data protection in graph data under differential privacy. In this regard, I aim to develop a personalized privacy-preserving mechanism for graph data. The challenge is to anonymize network data distributions under personalized differential privacy (PDP) such that different individuals might have different privacy levels based on their own preferences. A desired solution should be able to meet the following requirements:

- Provide freedom to individuals (nodes) to decide their own privacy preference (i.e., degree of privacy protection), which should be taken into consideration while releasing sensitive information of an individual.
- Preserve privacy while enhancing output utility under personalized (edge or node) differential privacy.
- Enhance output utility by reducing the amount of noise needed to achieve personalized differential privacy.

As each node in a network has its own privacy preference in a personalized setting, it is challenging to balance privacy and utility because a higher privacy level can add more noise. Additionally, each node has its own privacy preference whereas each data point in data distribution reflects information about more than one node. Moreover, reducing sensitivity is more challenging under node-DP than for edge-DP as graph data is highly sensitive to structural changes under node-DP.

## 1.4 Contributions

This thesis primarily focuses on developing privacy-preserving data publishing mechanisms that can provide the privacy guarantees of differential privacy while enhancing utility of published data. The goal is to explore two areas of research: (1) relational data publishing (Part I), where I aim to publish differentially private datasets while enhancing the utility of published datasets by reducing the amount of noise needed to achieve differential privacy, and (2) graph data publishing (Part II), where I aim to publish graph data under the privacy guarantees of edge, node and personalized differential privacy while enhancing the utility of published data. The main contributions of this thesis are summarized as follow,

### 1.4.1 Publishing Relational Datasets with DP

To accomplish the first objective, I develop a framework for publishing differentially private datasets while enhancing published data utility, presented in Chapter 4. Given a query $f$ and a differential privacy parameter $\varepsilon$, there are two ways to enhance the utility of the response of the query $f$: (1) use a noise addition scheme that adds less noise to the response of the query $f$; (2) replace a query $f$ by a modified query that approximates $f$ and has less sensitivity. In this thesis, I take the second approach. I observe that microaggregation can help to reduce sensitivity for improving data utility, and based on this observation I incorporate microaggregation into the differentially private data publishing scenario. Generally, a microaggregation algorithm consists of two phases: (a) Partition - similar records in a dataset are partitioned into the same cluster; (b) Aggregation - records in the same cluster are aggregated. More specifically, the original query $f$ with high sensitivity is transformed to an approximate query $f \circ \mathcal{M}$, which has lower sensitivity. However, an arbitrary microaggregation algorithm cannot reduce sensitivity when being incorporated into differential privacy. To address this limitation, I present a novel microaggregation-based framework called the *stable microaggregation*, that can ensure that at most two pairs of corresponding clusters in microaggregated neighboring datasets can differ in a single record. I further develop a simple yet effective algorithm under the proposed framework. The experiments show that proposed framework outperforms the state-of-the-art methods in terms of data utility by reducing sensitivity significantly.

Then, in Chapter 5 I extend the above preliminary work and propose a *general* framework. The general framework introduces a parameter $\alpha$ to enforce that at most $\alpha$ pairs of corresponding clusters in microaggregated neighboring datasets can differ in a single record. Thus, the amount of noise added to differentially private datasets can always be controlled by the parameter $\alpha$ and stable microaggregation becomes a special case of the general framework with $\alpha = 2$. Additionally, $\alpha$ indicates the trade-off between errors introduced by achieving microaggregation and differential privacy. I present two novel algorithms under the general framework which are based on a partial order of elements (records) in a dataset. The experiments show that the proposed framework outperforms the sate-of-the-art methods by providing better within cluster homogeneity and also reducing noise added into differentially private datasets significantly, regardless of the size of a dataset.

### 1.4.2 Publishing Graphs with Edge-DP

To deal with the second objective, I aim to anonymize graphs under edge differential privacy, where edge-DP aims to hide the presence and absence of a single edge. Due to the high sensitivity of graph data on structural changes, the utility of anonymized graphs is severely limited. I observe that the dK-graph model [71; 72] for analyzing network topologies can serve as a good basis for generating structure-preserving anonymized graphs. Essentially, the dK-graph model generates dK-graphs by retaining a series of network topology properties being extracted from $d$-sized subgraphs in an original graph. In order to reduce the amount of random noise without compromising $\varepsilon$-differential privacy, I incorporate microaggregation techniques [21] into the dK graph model to reduce the sensitivity of queries. This enables us to apply perturbation on network topology properties at a flexible level of granularity, depending on the degree of microaggregation. More specifically, I present a novel framework, called *dK-microaggregation*, that can leverage a series of network topology properties to generate $\varepsilon$-differentially private anonymized graphs, presented in Chapter 6. Then, I propose a distance constrained algorithm for approximating a graph via microaggregation within the proposed framework, which enables us to reduce the amount of noise being added into $\varepsilon$-deferentially private anonymized graphs. The experiments validate the noise reduction of our proposed framework and show that our algorithm can effectively enhance the utility of generated anonymized graphs in comparison with the state-of-the-art methods.

### 1.4.3 Publishing Higher-Order Graph Statistics with Node-DP

To deal with the third objective, I aim to publish higher-order graph statistics under node-DP, where node-DP aims to hide the presence and absence of a single node and the set of edges incident to that node. I observe that $dK$-distributions [72; 71] can serve as a good basis for representing higher-order graph statistics. Essentially, $dK$-distributions capture connectivity of a network, and also contain useful information about subgraph-based and degree-based characteristics at multiple levels of granularity in a network. To explore the sensitivity of higher-order network statistics under node-DP, I theoretically analyze the sensitivity of $dK$-distributions for $d = 2$, i.e., *joint degree distribution* [73]. Further, in order to effectively control sensitivity under node-DP, I present a new graph projection technique which generates $\theta$-bounded graphs by applying a two-level ordering strategy. The motivation behind graph projection is to bound the sensitivity of publishing network statistics through a control on node degrees. The experiments have verified the utility enhancement and privacy guarantee of the proposed framework. To the best of my knowledge, this is the first study to publish higher-order graph statistics under node differential privacy, while enhancing graph data utility.

### 1.4.4 Publishing Network Data Distribution with Personalized-DP

I further incorporate personalization into differential privacy mechanism to accomplish the fourth objective. More specifically, in this thesis I show how to publish network data distributions (e.g., such as degree distribution and joint degree distribution) in a personalized differentially private manner under edge-DP and node-DP, where individuals (nodes) in a network can specify their own privacy preferences, as presented in Chapter 8. To the best of my knowledge, there is no work which considers both variations of differential privacy while considering personalization. As discussed before, graph data is highly sensitive to structural changes under differential privacy. To address this limitation, I introduce degree queries for controlled sensitivity, and theoretically analyze the sensitivity of these queries for publishing network data distributions under edge-DP and node-DP. This enables us to reduce the overall noise needed to achieve personalized (edge or node) differential privacy, thus enhancing utility significantly. Further, I introduce four approaches for generating personalized differentially private network data distributions while enhancing utility. The experiments show that the proposed approaches guarantee differential privacy and enhance output utility while taking personalization into account. To the best of my knowledge, this is the first study to publish network data distributions under personalized differential privacy, while enhancing network data utility.

## 1.5 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 provides a comprehensive literature review of privacy-preserving data publishing under relational data and graph data by outlining strengths and weaknesses of the existing relational and graph data anonymization approaches. Chapter 3 presents notations and definitions that will be extensively used throughout this thesis. Further, this thesis is structured in two parts: Part I consists of Chapter 4 and Chapter 5, which study the problem of relational data publishing with the aim of generating differentially private datasets while enhancing published data utility, Part II consists of Chapters 6 - 8, which explore graph data publishing with the aim of achieving differential privacy while enhancing output utility on graph data. More precisely, Chapter 6 studies the problem of publishing differentially private graphs under edge differential privacy, Chapter 7 studies the problem of publishing higher-order graph statistics under node differential privacy, and Chapter 8 studies the problem of incorporating personalization in the differentially private computation by publishing network data distributions under personalized differential privacy. Chapter 9 concludes the thesis and outlines interesting research directions for future work.

Each of Chapters 4-8 starts with an overview of the specific problem studied in the chapter. Then the framework and the algorithms for solving the problem is presented. Each chapter finishes with experimental results and a summary of the chapter.

Chapter 2

# Literature Review

This chapter summarizes the background and related work that contributes to the understanding of the techniques, principles, and concepts used to address the privacy-preserving data publishing problems in this thesis.

## 2.1 Relational Data Publishing

Data analysis has been widely performed in real-life applications. For instance, election networks are studied to discover different opinions in a community, and co-author networks are used to understand collaboration relationships among researchers. In the relational data setting, such data is stored in a table and each row (record) corresponds to a distinct individual. A table has a number of attributes that can be divided into the following four types [7]:

1. *Explicit Identifiers*: Attributes that can directly identify an individual, such as social security number, name, and driving license number.
2. *Quasi-Identifiers*: Attributes that can potentially identify an individual, such as date of birth, postal codes, profession, gender, race and ethnicity.
3. *Sensitive Attributes*: Attributes that can have sensitive information of an individual such as disease, salary, and occupation.
4. *Non-Sensitive Attribute*: All other attributes that do not fall into the previous three categories.

### 2.1.1 Types of Disclosure in Relational Data

As publishing data can pose a privacy threat, this would intrude upon individual privacy. Therefore, before sharing data, it is necessary to prevent the disclosure of sensitive information of an individual. The disclosure types in relational data can be broadly grouped into two categories [24; 60; 63]:

1. *Identity Disclosure*: When an individual is linked to a particular record in a released table and an invader can recognise an individual by matching quasi identifiers values.

2. *Attribute Disclosure*: When new information about some individuals is revealed and an invader is able to determine some new characteristics of an individual based on released data.

Once there is identity disclosure occurring, an individual is re-identified and the corresponding sensitive values are revealed. Identity disclosure often leads to attribute disclosure; however, attribute disclosure can occur with or without identity disclosure [63].

### 2.1.2 Types of Background Knowledge in Relational Data

A fundamental conjecture for the disclosure risk is that there in an invader who has some background knowledge. The background knowledge is described by a set of variables which can be the attributes of a released table. One assumption in privacy-preserving data publishing (PPDP) is that a data recipient could be an invader who has three types of information:

1. The actual dataset that has been published.
2. Some information about individuals in this published dataset (e.g., post code).
3. Other publicly available data (e.g., census data).

An invader can use the available background knowledge to reveal the identity of an individuals and disclose the sensitive data.

### 2.1.3 Relational Data Anonymization Approaches

Data in its original form holds sensitive information about individuals, and publishing such data would intrude upon individual privacy. Thus, to preserve data privacy of individuals, various anonymization techniques [92; 67; 63; 102; 100; 113; 25] have been proposed for relational data publishing. Figure 2.1 provides a general overview of the main relational data anonymization approaches, which are discussed below.

#### 2.1.3.1 *k*-Anonymity and Related Approaches

Here I present an overview of *k*-anonymity and its related approaches for privacy-preserving data publishing.

***k*-Anonymity.** Sweeney introduced *k-anonymity* [92; 93] that ensures *identity disclosure*, and required every record in a dataset is indistinguishable among $k$ - 1 others records in terms of quasi-identifying attributes. A data release provides $k$-anonymity protection if the information for each individual contained in the release cannot be distinguished from at least $k$ - 1 individuals whose information also appears in the release. In simple words, a table is said to be $k$-anonymous if there exist at least $k$ - 1 records whose quasi-identifiers have the same values. Such records with the same values of quasi-identifiers are referred to as an *equivalence class*. Sweeney also

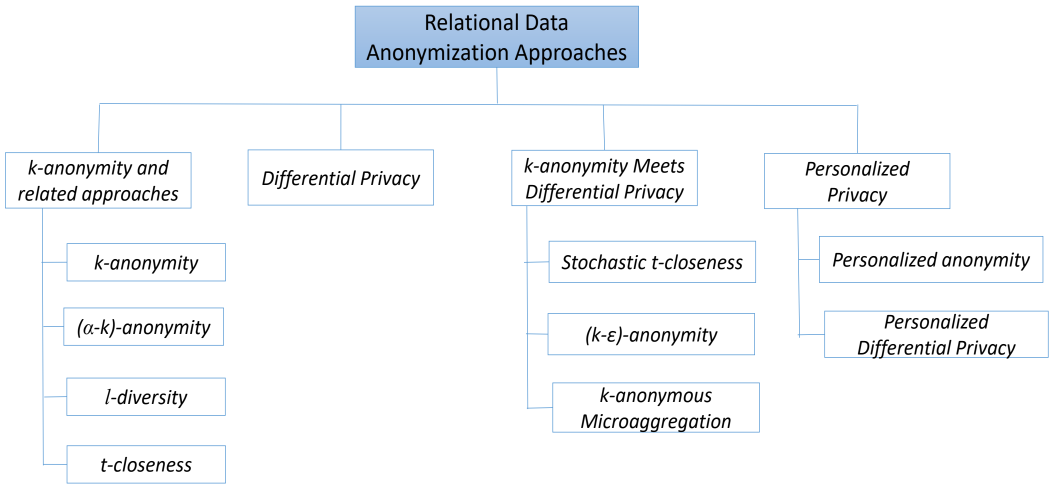

Relational Data Anonymization Approaches
k-anonymity and related approaches
Differential Privacy
k-anonymity Meets Differential Privacy
Personalized Privacy
k-anonymity
(α-k)-anonymity
l-diversity
t-closeness
Stochastic t-closeness
(k-ε)-anonymity
k-anonymous Microaggregation
Personalized anonymity
Personalized Differential Privacy

**Figure 2.1**: Taxonomy for relational data anonymization approaches.

examined some attacks on the proposed $k$-anonymity model and provided solutions to these attacks.

- *Unsorted Matching Attack:* This attack is based on the order in which rows appear in a published table. If a table is released under $k$-anonymity, and a subsequent release of the same table is then performed, then direct matching of rows across the tables based on row position within the tables revealed sensitive information. The author has provided a simple solution to this attack by performing random sorting on the rows of the $k$-anonymous tables.

- *Complementary Release Attack:* This attack is based on releasing different $k$-anonymous tables of an original table and by linking these $k$-anonymous tables one can reveal the original table. This attack can be prevented if there is no quasi-identifier value more specific than what appears in the first released table would appear in the subsequent tables.

- *Temporal Attack:* This attack occurred in dynamic data collection circumstances where rows are added, changed, and removed constantly. As a result, data released over time can be subject to a temporal inference attack. To prevent this attack, either all of the attributes of the first released table would be considered as quasi-identifiers for subsequent releases, or subsequent releases themselves would be based on the first released table.

**($\alpha$, $k$)-Anonymity.** The $k$-anonymity privacy model guarantees the privacy of individuals' data before it is released by a data publisher. But it did not protect the association of individuals to sensitive information. In the work [100], the authors extended the $k$-anonymity model to protect both individual identification and relationships to sensitive information in published data. The general idea is to require a percentage of records containing sensitive values in the equivalence class should be less than $\alpha$, where $\alpha$ is a threshold specified by a data publisher. Nonetheless, if the values of sensitive attributes are skewed, then ($\alpha$, $k$)-anonymity may result in high distortion [4] and provide less data utility.

**$l$-Diversity.** The notion of *$l$-diversity* [67] has been proposed to address the limitations of $k$-anonymity. Two attacks have been identified over $k$-anonymity in this work.

- *Homogeneity Attack:* An invader can find out the values of sensitive attributes when there is less diversity in them.
- *Background Knowledge Attack:* An invader can have some background knowledge on which inferences can be drawn about sensitive information.

To prevent these two attacks, the definition of $l$-diversity is formulated as follow: the sensitive values in every equivalence class (a set of records in a $k$-anonymized table which have the same values for quasi-identifiers) of sensitive attributes must be diverse (less uniform). The *Bayes-optimal Privacy* is used as a starting point for a definition of privacy. It involves modeling the background knowledge as a probability distribution over attributes and uses the Bayesian inference technique. The prior belief and posterior belief of an invader have been calculated to define the privacy principle, i.e., there should not be a large difference between the prior belief and posterior belief of the invader with some background knowledge. Moreover, $l$-Diversity ensures the diversity values in a sensitive attribute. Despite considerable progress, $l$-diversity is still insufficient to prevent attribute disclosure if values in sensitive attribute are naturally more frequent than others in an equivalence class. This may enable an invader to infer that a record in an equivalence class is very likely to have these sensitive values. On the other hand, in [22] it is shown that enforcing $k$-anonymity with $k = l$ may lead to high information loss and decreased utility of the published data.

**$t$-Closeness.** The notion of *$t$-Closeness* [63] has been proposed to address the limitations of $l$-diversity. Two attacks have been identified over $l$-diversity in this work.

- *Skewness Attack:* Attribute disclosure occurs when an overall distribution is skewed.
- *Similarity Attack:* When values in sensitive attributes are semantically similar.

Moreover, it has been identified that, it is difficult and unnecessary to achieve $l$-diversity. *$t$-closeness* addresses the limitations of $l$-diversity. An equivalence class is said to have $t$-closeness if the distance between two distributions, i.e., the distribution of a sensitive attribute in any equivalence class and the distribution of the corresponding attribute in an original table, should be no more than a threshold $t$. A table is said to have *$t$-closeness* if all equivalence classes have t-closeness. Nevertheless, $t$-closeness degrades the data utility because $t$-closeness destroys the correlations between quasi-identifiers and sensitive attributes. One way to reduce the damage is

**Table 2.1**: $k$-anonymity and related approaches with disclosure types.

| Approaches | Identity Disclosure | Attribute Disclosure |
|---|---|---|
| $k$-Anonymity | ✓ | ✗ |
| $(\alpha, k)$-Anonymity | ✓ | ✓ |
| $l$-Diversity | ✓ | ✗ |
| $t$-Closeness | ✓ | ✓ |

to adjust the thresholds with the increased risk of skewness attack [22], or employ differential privacy which provides a guard against skewness attack [4].

**Discussion.** In a nutshell, $k$-anonymity ensures identity disclosure but cannot prevent attribute disclosure. Its extension, $l$-diversity, tried to overcome this limitation; however, it is not sufficient to prevent attribute disclosure because of *skewness attack*. Another enhancement of $k$-anonymity is $(\alpha, k)$-anonymity which ensures both identity disclosure and attribute disclosure but provides high distortion when sensitive values are skewed. Additionally, $t$-closeness was the one which can protect microdata release from both identity disclosure and attribute disclosure but damage the correlations between quasi-identifiers and sensitive attributes. Table 2.1 summarizes the information disclosure types being addressed by the anonymization approaches described in this section. To overcome the issues related to $k$-anonymity and its related extensions, in the work [29; 25], the notion of *differential privacy* was introduced.

#### 2.1.3.2 Differential Privacy

Anonymization approaches such as $k$-anonymity and its extensions guarantee privacy by enforcing syntactic restrictions on data output. For instance, a data output is needed to be indistinguishable among $k$ records, or sensitive values are represented in every equivalence class as discussed in the previous section. These privacy definitions have been criticized due to the feasibility of the attacks [100; 22; 113]. In [29] a privacy model called *differential privacy* was proposed. Differential privacy is a mathematical definition for privacy loss when private information of an individual is released. Alternative to $k$-anonymity and its extensions, differential privacy essentially guarantees the output of a randomized mechanism will be almost indistinguishable with or without any single individual's information included in a dataset $X$. Therefore, from the perspective of an individual, a data output is computed as if from a dataset that does not hold an individuals' record.

The term was devised by [25], but the correct reference is actually an earlier publication by [29]. Differential privacy ensures that the output of a computation over any two neighboring datasets (i.e., datasets that differs in a single record) induces almost similar distributions vary by a factor $e^{\varepsilon}$, where $\varepsilon$ is a privacy budget, as shown in Figure 2.2. Smaller values of $\varepsilon$ provide stronger privacy guarantees [28]. Thus, an invader cannot infer the presence or absence of a single record in an input based on any other output, regardless of any background information available. Differen-

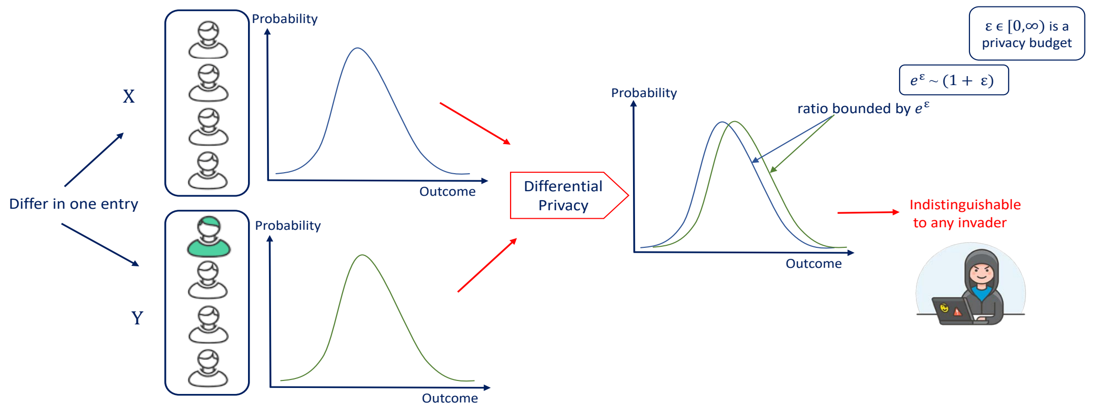

Probability
Outcome
X
Y
Differ in one entry
Differential Privacy
ratio bounded by $e^{\varepsilon}$
$e^{\varepsilon} \sim (1 + \varepsilon)$
$\varepsilon \in [0,\infty)$ is a privacy budget
Indistinguishable to any invader

**Figure 2.2**: An overview of differential privacy.

tial privacy was originally proposed as a privacy model to protect the responses of interactive queries to a dataset. Let $f$ be an arbitrary query function for which a differentially private response is requested. A typical differential privacy mechanism works as follow:

- Given a query $f$, compute a real response $f(X)$.
- Output a masked response $f(X) + N(X)$, where $N(X)$ is a random noise calibrated according the sensitivity $\Delta$ of $f$.

To generate $N(X)$, a common choice is to use a Laplace distribution with zero mean and a scale parameter $\Delta(f)/\varepsilon$, where:

- $\varepsilon$ is the differential privacy parameter/level/budget;
- $\Delta(f)$ is the L1-sensitivity of $f$. That is, the maximum variation in the query function $f$ between neighboring datasets, i.e., datasets differing in at most one record.

Note that, for fixed values of the privacy parameter $\varepsilon$, the higher the sensitivity $\Delta(f)$ of the query function $f$, the more noise is added. Indeed, guaranteeing the differential privacy requires more noise when the query function $f$ can vary strongly between neighboring datasets. Moreover, for a fixed $\Delta(f)$, the smaller $\varepsilon$, the more noise is added. The privacy is attained at the cost of accuracy by adding random noise to the data [44], and there is a always a trade-off between privacy and utility.

**DP based approaches.** With the emerging of differential privacy in recent years [29; 25], a number of works considered releasing differentially private datasets [68; 105]. Such differentially private datasets can guarantee differential privacy controlled by a privacy parameter $\varepsilon$ in a robust statistical way. Many approaches [86; 87; 90] used differential privacy to generate differentially private datasets in the relational data setting, where a dataset output by a differential privacy mechanism is denoted as a differentially private dataset. In the works [86; 87; 90; 91], differentially private datasets are generated via record masking, in which the original query was approximated to reduce the sensitivity of queries used to generate differentially private datasets.

**Challenges and Limitations.** In recent years, differential privacy emerges as a significant and rigorous mathematical definition of privacy [27], however, one of the key challenges in differential privacy is to preserve data privacy while achieving data utility. Broadly speaking, there are two common methods used for generating $\varepsilon$-differentially private datasets in the literature: one is based on differential privacy compliant histograms [105] and the other is based on record perturbation [90]. Histogram-based approaches have some limitations, including: being limited to histogram queries and the exponential growth of the number of histogram bins with the number of attributes [89]. On the other hand, record perturbation based approaches require a large amount of noise added to the results of queries and analyses [91], though these approaches are not limited to histogram queries and allow dealing with any type of attributes.

Nevertheless, when generating differentially private datasets, there always is a trade-off being made between privacy and utility of published data. Ideally, we want to preserve the privacy of individuals while still maintaining the usefulness of data for performing statistical analysis. The utility of $\varepsilon$-differentially private datasets is however limited due to the amount of noise being added to guarantee differential privacy.

**Application Domains.** Differential privacy can be applied to many real-world application domains, and produce anonymized data that can preserve privacy of an individual where private data analytics are used in practice. One real-life application is social networks, where differentially private techniques can be used to ensure the privacy of individuals when publishing insights about social network activities [5; 37; 41; 50; 54]. In health sector differential privacy has been proposed as a possible approach to allow the release of healthcare data with sufficient guarantee against possible privacy attacks [31; 55]. Financial sector is another common domain where differential privacy allows businesses such as banks and insurers to model customer behavior, without violating their privacy, leading to tailored products and services [40; 82]. In a nutshell, differential privacy helps in various industrial and commercial sectors to perform private data analysis without breaching individuals privacy.

#### 2.1.3.3 *k*-Anonymity Meets Differential Privacy

Below I present an overview of relational data anonymization approaches which are based on synergy between *k*-anonymity and differential privacy.

**Stochastic *t*-Closeness.** Before the work [20], it was not straightforward to satisfy *t*-closeness for a dataset that is differentially private because differential privacy is stochastic, whereas *t*-closeness is deterministic. However, in [20], the authors extended *t*-closeness [63] by relating it with differential privacy and proposes a novel privacy technique called *stochastic t-closeness*. Stochastic $t$-closeness provides differential privacy when $t = exp(\varepsilon/2)$. In stochastic $t$-closeness, to satisfy $t$-closeness for sensitive attributes, a stochastic function is used in place of an empirical distribution, as used in classic $t$-closeness.

**$(k, \varepsilon)$-Anonymity.** To guarantee the privacy of an individual, a privacy-preserving data publishing framework [43], called $(k, \varepsilon)$-Anonymity, was proposed by combining $k$-anonymity and $\varepsilon$-differential privacy. In this approach, attributes of a dataset have been classified into three categories: explicit identifiers, quasi identifiers and sensitive attributes. Further, quasi-identifiers have been divided into two categories: $k$-quasis and $\varepsilon$-quasis. $k$-quasis can be any quasi attribute but $\varepsilon$-quasis must be a numerical quasi attribute. Afterward, $k$-anonymity is applied on $k$-quasis which divides a dataset into $k$ equivalence classes. Then $\varepsilon$-differential privacy is applied on $\varepsilon$-quasis. Then, the records are shuffled in a dataset to prevent the order-based attacks in the published data. However, how to divide a dataset into $k$-quasis and $\varepsilon$-quasis among quasi-identifiers was not discussed.

**Differential Privacy via $k$-anonymous Microaggregation** A number of works [91; 89] have combined $k$-anonymity and differential privacy to enhance the utility of data release. One of these works used microaggregation to achieve $k$-anonymity, which can reduce the amount of noise added to differentially private datasets [21]. Microaggregation [18] is a family of anonymization algorithms that work in two stages:

1. *Partitioning:* Generate clusters of records in a dataset in such a way that each cluster comprises of at least $k$ records and records within each cluster are homogeneous (as similar as possible).
2. *Aggregation:* Each record within each cluster is replaced by its centroid record (average record).

For multivariate records (two or more attributes), optimal multivariate microaggregation, i.e., with minimum variability loss within clusters, is an NP-hard problem [80], so heuristics are normally used. Numerous heuristic techniques to multivariate microaggregation have been proposed in the literature, and two main types of heuristics [19] are below:

1. *Fixed-size microaggregation* yields clusters with a fixed size $k$, where all clusters have the same size $k$, except one cluster which is of size between $k$ and $2k - 1$. These heuristics are computationally efficient.
2. *Data-oriented microaggregation* yields clusters with variable sizes, where all clusters have sizes varying between $k$ and $2k - 1$, which may achieve lower information loss [18].

In the work [21], it is shown that $k$-anonymity can be achieved via multivariate microaggregation, and a simple microaggregation heuristic called *MDAV* was proposed which groups a dataset into exactly $k$ clusters except the last one, which contains records between $k$ and $2k-1$. Later, in [90; 91], multivariate microaggregation is used to generate differentially private datasets by querying the centroid (arithmetic average) of each cluster. One can regard the approach of multivariate microaggregation as combining $k$-anonymity and differential privacy because with minimal cluster size $k$, multivariate microaggregation over all quasi-identifiers ensures $k$-anonymity. The target of a microaggregation algorithm is to yield minimum information loss by maximizing within cluster homogeneity. However, the existing works, including MDAV [21] and insensitive microaggregation [91], either produce a low degree of within cluster homogeneity or fail to reduce the amount of noise independent of the size of a dataset.

#### 2.1.3.4 Personalized Privacy

Personalized privacy provides freedom to individuals to decide their own privacy preference (i.e., degree of privacy protection), which should be taken into consideration while releasing sensitive information of an individual. Early privacy-preserving data publishing techniques [92; 67; 63; 25] focused on protecting individuals' private information by using predefined constraint parameters specified by data holders, which may result in offering insufficient protection to a subset of individuals, while applying excessive privacy control to others. In the personalized setting a person can specify the degree of privacy protection to guard their sensitive information which helps in the data release process to provide sufficient amount of privacy without under or over protecting an individual. For example, for privacy purposes an individual may not be interested in sharing his/her telephone number; however a retail organization might need to share their number for the convenience of their customers. Therefore, privacy requirements are different for different individuals, and individuals should also have the right and responsibility to protect their own private information. Thus, there is a need for personalized privacy-preserving mechanisms which are intuitive for novice users. Following are some relational data anonymization approaches based on personalization.

**Personalized Anonymity.** In the work [102] the notion of personalized privacy was proposed for $k$-anonymity [92]. Personalized anonymity allows each individual to specify her own privacy level based on its own tolerance to sensitivity. The personalized anonymity model assumes that each sensitive attribute has a taxonomy tree and that each individual specifies a guarding node in this tree. The advantage of this approach is that each individual may specify a guarding node according to their own preference. It is shown that personalized anonymity result in lower information loss than the well-known privacy models, such as $k$-anonymity [92] and $l$-diversity [63]. However, it is unclear how individuals would set their guarding node because guarding node depends on the distribution of sensitive values in the whole table or in a group which is unknown to an individual because an individual has no access to the distribution of sensitive values before the data is published. Without such information, an individual may select a more general (higher privacy preference) guarding node, which may negatively affect the utility of data. Later, the concept of personalization is used to extent the notion of $(\alpha, k)$-anonymity [100] and personalized $(\alpha, k)$ model was presented in the work [110]. The authors proposed an efficient anonymization algorithm based on $(\alpha, k)$-anonymity by introducing a vector for describing individuals' personalized privacy preference. The general idea is to combine $(\alpha, k)$-anonymity [100], which provides protection against homogeneity attack, and personalized privacy techniques which provide freedom to individuals to specify their own degree of privacy. This improves data utility while satisfying different individuals privacy preserving requirements.

**Personalized Differential Privacy (PDP).** One important limitation of differential privacy [29] is that the same privacy protection level (i.e., $\varepsilon$) is assigned to all individuals while adding noise. However, different individuals have different privacy levels based on their own tolerance to sensitivity subject to their data. Therefore, differential privacy may lead to provide insufficient privacy protection for some individuals, while over-protecting others.

Relational data anonymization approaches based on differential privacy [29] where an individual can set their own privacy budget $\varepsilon$ can be broadly categorised into two types: user-grained [32; 49; 62], and item-grained [2] personalized differential privacy.

- *User-grained:* This approach was first presented in [32] which generalized the classic definition of differential privacy to provide freedom to each individual to have a personalized privacy budget $\varepsilon$ based on their personal privacy preferences called personalized differential privacy (PDP). Similarly, in the work [49] the authors proposed mechanisms based on personalization to achieve user-grained differential privacy. However, these mechanisms either under-utilize the privacy budget of most individuals or require a careful design in order to get the best results in terms of accuracy and scalability for achieving PDP. In another work [62] two partitioning-based mechanisms were proposed to achieve user-grained PDP in which a dataset is partitioned based on the privacy preference provided by an individual. Then, applying differential privacy mechanism on each partition using the strongest privacy protection level.

- *Item-grained:* In item-grained differential privacy, an individual is allowed to set different privacy protection levels for different data items (e.g. salary, age, etc). For relational data the first item-grained approach was proposed in [2] which is called heterogeneous differential privacy (HDP).

Table 2.2: Summary of relational data anonymization approaches.

| Relational Data Anonymization Approaches | | Main Idea | Pros/Cons |
|---|---|---|---|
| $k$-Anonymity and Related Approaches | $k$-Anonymity | Hides individuals' data in $k-1$ other individuals. | Pros: Ensure identity disclosure.<br>Cons: Cannot prevent attribute disclosure. |
| | $(\alpha, k)$-Anonymity | Protect both individual identification and relationships to sensitive information. | Pros: Ensure both identity and attribute disclosure.<br>Cons: Provide high distortion when sensitive values are skewed. |
| | $l$-Diversity | Ensure sensitive attributes must be diverse while achieving k-anonymity. | Pros: Ensure identity disclosure.<br>Cons: Cannot prevent attribute disclosure when some sensitive values are more frequent than others. |
| | $t$-Closeness. | Ensure that the distribution of sensitive values to be the same in all k-groups. | Pros: Ensure both identity and attribute disclosure.<br>Cons: Damage the correlations between sensitive attributes and other attributes. |
| Differential Privacy | | Provide mathematical privacy guarantee regardless of background knowledge. | Pros: Mathematical privacy guarantee.<br>Cons: Limit data utility. |
| $k$-Anonymity Meets Differential Privacy | Stochastic $t$-Closeness | Satisfy $t$-closeness for a dataset that is differentially private. | Pros: Provide differential privacy.<br>Cons: Need careful design. |
| | $(k, \varepsilon)$-Anonymity | Satisfy differential privacy over $k$-anonymous data. | Pros: Provide differential privacy.<br>Cons: How to divide a dataset among quasi-identifiers is unclear. |
| | $k$-anonymous Microaggregation | Group similar (homogeneous) records into clusters, and then replace each record with its cluster representative. | Pros: Enhance data utility.<br>Cons: Optimal microaggregation is an NP-hard problem. |
| Personalized Privacy | Personalized Anonymity | An individual can specify the degree of privacy protection to guard his/her sensitive information. | Pros: Provide personalization.<br>Cons: Difficult to design. |
| | Personalized Differential Privacy | Non-uniform $\varepsilon$ can be used to achieve differential privacy. | Pros: Provide personalization.<br>Cons: Difficult to achieve. |

#### 2.1.3.5 Summary

Until now, a number of relational data anonymization approaches have been proposed to limit disclosure of confidential information. For clarity, Table 2.2 summarises the main idea of each relational data anonymization approach and highlights their pros and cons.

## 2.2 Graph Data Publishing

Graph dataset storage and management is an active area of research in both academia and industry, as evidenced by the growing number of graph database systems in real-world applications such as social networks. In a network, nodes correspond to individuals or entities, and edges correspond to relationships between them. However, graph data may contain sensitive, private or personally identifiable information of individuals and directly releasing graph datasets for analysis may leak sensitive information of an individual. As more and more networks have been made publicly available, preserving privacy in publishing graph data becomes an important task. A recent study [3] shows that simple techniques of anonymizing graphs by removing the identities of nodes before publishing graph data does not always guarantee privacy.

### 2.2.1 Types of Disclosure in Graph Data

The privacy risks or disclosure types in publishing graph data can be broadly grouped into three categories [66; 1; 78]:

1. *Identity Disclosure:* The identity of individuals associated with graph nodes is breached.
2. *Membership Disclosure:* Sensitive relationship information among individuals associated with graph edges is revealed.
3. *Content Disclosure:* Sensitive attributes associated with each node are disclosed.

The identity and membership disclosure attacks are usually related to the reason that the disclosure of a sensitive link requires the identification individuals (i.e., nodes) as an intermediary step. Both forms of privacy risks are related to the structural anonymization of a graph and do not assume that the sensitive attributes associated with each node is released along with a graph. Contrary, in the content disclosure attack, sensitive attributes associated with each node are always released with graph nodes, and an individual may not wish to expose this information. This form of privacy risks is related to the sensitive content anonymization of a graph.

### 2.2.2 Types of Background Knowledge in Graph Data

Since it is known that an invader can re-identify individuals easily from a published graph based on some background knowledge, and removing identity of nodes is not sufficient to preserve privacy of an individual [81; 48]. An invader may obtain different kinds of background knowledge about a target individual. There are three types of background knowledge analytically captured by Hay *et al.* [41]:

1. *Vertex knowledge* refers to the knowledge associated with local structure of the graph around a node (e.g., node degree).
2. *Subgraph knowledge* describes the knowledge about (partial) subgraphs (e.g., triangle, star, $d$-hop neighbors) around nodes.
3. *Hub fingerprint knowledge* alludes to the knowledge related to nodes and their distances from hub nodes.

Some existing works [115; 116] have considered very strong invaders equipped with subgraph or hub fingerprint knowledge. It was highlighted in [3] that in real social networks, most nodes belong to a small uniquely distinguishable subgraph; hence, it is comparatively easy for an invader to obtain subgraph background knowledge related with a node to conduct an attack.

### 2.2.3 Types of Structural Attacks in Graph Data

The structure of a graph itself, and in its basic form such as degree of the nodes, can be reveal the identities of individuals [41]. Structural attacks in graph data publishing are categorised into four types [10; 115; 66]:

1. *Degree Attack:* An invader equipped with vertex knowledge and a subgraph query Q can uniquely identify a node based on degree of a target node, where partial information about a node can be expressed as a subgraph query Q (e.g., number of neighbors of a node).
2. *Subgraph Attack:* An invader armed with subgraph knowledge and a subgraph query Q can uniquely identify a target node based on the matches of Q in a published network.
3. *Neighbor-Graph Attack (NAG):* An invader provided with $d$-hop neighbors information of a target node and a subgraph query Q can uniquely identify a target node based on the matches of Q in a published network.
4. *Hub Fingerprint Attack:* An invader supplied with hub fingerprint knowledge and a subgraph query Q can uniquely identify a target node based on the known distances between a target node and hubs.

### 2.2.4 Graph Data Anonymization Approaches

A number of anonymization methods and algorithms for graph data have been proposed in the literature [41; 66; 9; 41; 114; 14; 50], which can be categorized into approaches based on *k-anonymity*, *graph modification*, *generalization or clustering*, and *differential privacy*. Figure 2.3 provides a general overview of the graph data anonymization approaches, which are discussed below.

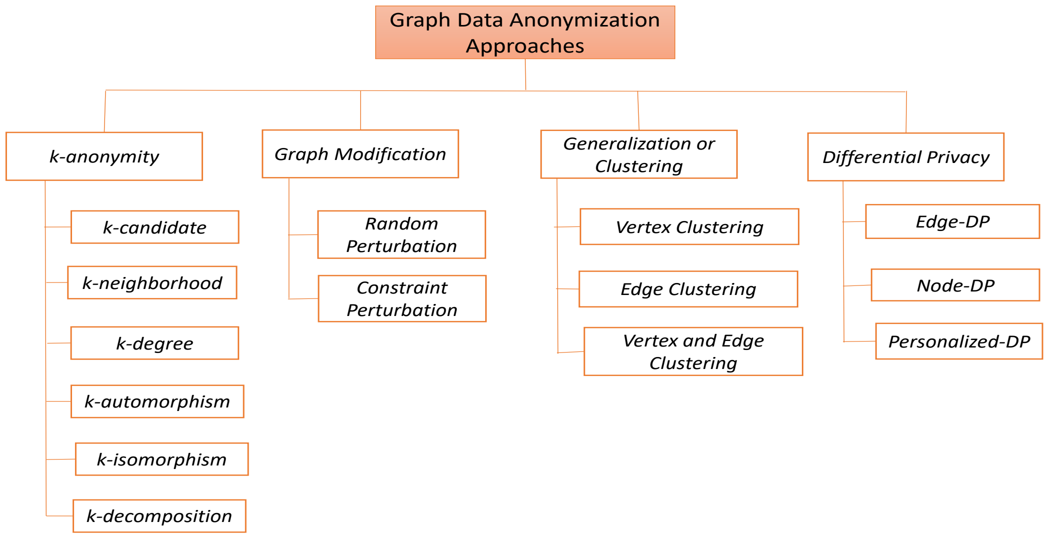


**Figure 2.3**: Taxonomy for graph data anonymization approaches

#### 2.2.4.1 $k$-Anonymity Based Approaches

Generally, $k$-anonymity approaches divide an original graph into at least $k$ blocks, so that the results of a query on the original graph are at least $k$ parts, and the probability that an invader can re-identify one node's identity is at most $1/k$. Here I present an overview of $k$-anonymity based graph data annonymization approaches.

**$k$-Candidate Anonymity.** Hay *et al.* [42] proposed to protect privacy against subgraph knowledge and highlighted that the structural similarity of neighbors of nodes in a graph determines the extent to which an individual in a graph can be re-identified. Thus they presented a notion of $k$-candidate: a graph satisfies $k$-candidate anonymity if for every query over a graph, there exist at least $k$ nodes that match the query. The authors generally focused on providing a set of anonymity requirements and studying their properties. However, they did not present algorithms that guarantee the construction of a graph that satisfies their anonymity requirements.

**$k$-Neighborhood Anonymity ($k$-NA).** Zhou and Pei [115] provided a solution against 1-neighborhood attack (NAG). They assumed that an invader is aware of a subgraph constructed with 1-neighborhood subgraph of a target node. They proposed an algorithm that organizes nodes into groups, according to their neighborhoods, and then anonymizes the neighborhoods of nodes in the same group. The authors generally focused on 1-neighborhood subgraph which is later generalized by the work [116]. In this work, the idea of $k$-automorphism [116] was proposed where an invader knows any subgraph around a target node ($d$-neighborhood subgraph).

**$k$-Degree Anonymity ($k$-DA).** Liu and Terzi [66] considered an invader equipped with vertex knowledge and proposed $k$-degree anonymity based on an edge addition/deletion technique. A published graph is $k$-degree anonymous if for every node

**Table 2.3**: *k*-anonymity based graph data anonymization approaches with disclosure types.

| Approaches | Identity Disclosure | Membership Disclosure | Content Disclosure |
|---|---|---|---|
| *k*-candidate | ✓ | ✗ | ✗ |
| *k*-NA | ✓ | ✓ | ✗ |
| *k*-DA | ✓ | ✗ | ✗ |
| *k*-auto | ✓ | ✗ | ✗ |
| *k*-iso | ✓ | ✓ | ✗ |
| *k*-dec | ✓ | ✓ | ✗ |

$v$ in a graph, there are at least ($k$–1) other nodes that have the same node degrees as $v$. The authors considered a graph with a degree sequence ordered in decreasing order and achieved $k$-anonymity by anonymizing the degree sequence first and then constructing an anonymized graph using an anonymized degree sequence. In this process, if the necessary condition did not fulfill for the anonymized degree sequence of a given graph, there does not exist an anonymized graph. Furthermore, if the value of $k$ is large and the graph is sparse, then it will degrade the utility of the anonymized graph. Additionally, this approach is not efficient on large graphs as for a given degree sequence it is not always possible to get an anonymized graph. They presented an idea of relaxing anonymized graph construction which leads to major modification of graph topological properties such as average betweenness, clustering coefficient, and average path length.

***k*-Automorphism (*k*-auto).** Zou *et al.* [116] presented a general framework in which they have adopted more general assumptions about an invader's background knowledge and considered a scenario where an invader knows any subgraph around a target node ($d$-neighborhood subgraph). In general, $k$-automorphism divides a large graph into $k$ blocks so that each subgraph corresponds to at least $k-1$ subgraphs, and there is at least one connecting edge among subgraphs. For achieving $k$-automorphic anonymity, the presented algorithm first partitions an original graph into $n$ blocks using the idea of frequent subgraph [58] which find $k$ matches of a subgraph in a given graph. After partitioning the graph into $n$ blocks it clusters the blocks into $m$ groups where each group has at least $k$-blocks using a proposed graph alignment algorithm. In the end, for crossing edges, it performs edge-copy to obtain an anonymized graph. The proposed method provides guard against structural attacks and anonymity for dynamic release of a graph. However, later in [10; 15] it has been shown that $k$-automorphism does not guarantee privacy for membership disclosure and is vulnerable under neighbor-graph attack (NAG).

***k*-Isomorphism (*k*-iso).** Cheng *et al.*[10] presented an idea of $k$-isomorphism ($k$-iso), which is an improved version of $k$-automorphism [116]. Unlike $k$-auto, $k$-iso fully divides an original graph into $k$ disjoint subgraphs in such a way that there is no connecting edge among subgraphs and subgraphs in each block are pairwise iso-

**Table 2.4:** *k*-anonymity based graph data anonymization approaches with structural attacks.

| Approaches | Degree Attack | Subgraph Attack | Neighbor-graph Attack | Hub Fingerprint Attack |
|---|---|---|---|---|
| *k*-candidate | ✗ | ✓ | ✗ | ✗ |
| *k*-NA | ✗ | ✗ | ✓ | ✗ |
| *k*-DA | ✓ | ✗ | ✗ | ✗ |
| *k*-auto | ✓ | ✓ | ✗ | ✗ |
| *k*-iso | ✓ | ✓ | ✓ | ✓ |
| *k*-dec | ✓ | ✓ | ✓ | ✗ |

morphic. For achieving *k*-isomorphism the presented technique first partitions a graph into *k* subgraphs with the same number of nodes. In order to partition a graph the proposed algorithm computes a set of potential anonymization subgraphs (PAGs) that covers an original graph by using frequent subgraphs with a given a node degree size (maximum degree). After finding a set of PAGs, the subgraphs are augmented by edge addition and deletion to ensure pairwise graph isomorphism. The proposed method also provides guard against structural attacks under neighbor-graph attacks and anonymity for dynamic release of a graph by simplifying the idea presented in *k*-auto [116]. As *k*-iso divides an original graph into *k* disjoint subgraphs, all the crossing edges need to be deleted which may contain some important and sensitive edges. Consequently, *k*-iso satisfies the *k*-anonymity principle at the expense of utility. Additionally, in order to ensure pairwise isomorphism when *k* is large and original graph is sparse, *k*-iso did not preserve the structural similarity between original graph and an anonymized graph.

***k*-Decomposition (*k*-dec).** Ding *et al.* [15], proposed a *k*-decomposition (*k*-dec) algorithm that is designed to anonymize large-scale graph datasets. This k-dec algorithm is based on k-automorphism (k-auto) and *k*-isomorphism (*k*-iso). *k*-dec first divides a graph into *k* disjoint blocks by using a multilevel *k*-way partitioning algorithm, then retain the deleted connections among *k* disjoint blocks using a proposed retain algorithm and achieves an anonymized graph, such that each *k* disjoint block is isomorphic to each other by using a proposed isomorphic algorithm. In order to retain cross edges the proposed algorithm adds noisy edges and nodes. Thus, to generate *k* disjoint isomorphic blocks, it needs to add a similar amount of nodes and edges into each block which overall results in adding too much noise in an anonymized graph. On the other hand, *k*-dec used a multilevel *k*-way algorithm to partition an original into *k* disjoint blocks by following the principle that the nodes and edges should be evenly divided into blocks. Thus, the quality of the anonymized graph is highly depended on the partition.

Table 2.3 summarizes the information disclosure types being addressed by the graph anonymization approaches described in this section. Additionally, Table 2.4 summarizes the structural attacks being guarded by the graph anonymization approaches described in this section.

#### 2.2.4.2 Graph Modification Based Approaches

Graph modification based approaches [9] anonymize a graph by modifying edges and nodes in a graph, i.e., new edges, and nodes are added or removed to ensure that published graph data meets desired privacy requirements. These graph modification methods can be categorized into approaches based on *random perturbation*, and *constraint perturbation*.

- *Random Perturbation:* Methods built on random perturbation follow certain principles to process nodes and edges, which includes *Rand add/del* and *Rand Switch* [42; 34; 94] that randomly add noise either by adding, removing, and switching edges or nodes, *Spctr add/del* and *Spctr Switch* [112] that are specifically designed to preserve the spectral characteristics of an original graph, *Blockwise Random add/del* [111] that divides a graph into blocks according to a degree sequence and implements edge modification on nodes at high risk of re-identification, and *Sparsification* [6] which randomly removes edges without adding new ones and uses entropy quantification to measure the level of anonymity provided by a perturbed graph.

- *Constraint Perturbation:* Approaches based on constraint perturbation modify a graph to meet some desired constraints. Perhaps, $k$-anonymity based approaches [42; 66; 116; 115] are the most well-known in this group, which provide anonymity by adding or deleting edges or nodes of a graph to meet some certain constant value $k$.

#### 2.2.4.3 Generalization or Clustering Based Approaches

Generalization methods also known as clustering-based methods anonymize graph data by classifying nodes and edges into groups. Then each group is generalized into the super nodes and super edges, along with some structural properties of the group. The generalized graph can hide the individual for preserving privacy, though, the graph may be shrunk after anonymization and local structures are difficult to analyze. However these anonymized graphs can be a good source to study the macro-properties of their original graphs. These graph generalization methods can be categorized into approaches based on:

- *Vertex clustering method* partitioned an original graph into disjoint sets where each super node represents at least $k$ nodes and each super edge represents all edges between nodes in two super nodes [41].

- *Edge clustering methods* partitioned an original graph wherein an edge (relational information) exists between clusters of nodes [114]. In [77], differential privacy is adapted to graph clustering and the notion of *m-edge-differential privacy* has been proposed. A Privacy-integrated graph clustering approach (*PIG*) has formalized for graph perturbation as noise adding mechanism to guarantee differential privacy.

- *Vertex and edge clustering methods* partitioned an original graph into clusters and then combined nodes in each cluster to form a super node and inter-cluster edges between clusters into a super edge [8]

#### 2.2.4.4 Differential Privacy Based Approaches

Differential privacy methods on graph data can be roughly divided into two categories, namely: node differential privacy (node-DP) [14] and edge differential privacy (edge-DP) [50]. Both edge-DP and node-DP are based on the single node or edge addition and deletion. However, since graph information affected by the addition and deletion of one node on the graph is always larger than the addition and deletion of an edge, node differential privacy technology is more challenging to design and implement [14]. These variants of differential privacy are used to perturb various statistical values of graph data, for instance triangle counting [16], degree distribution [14], and frequent subgraph mining [107]. In general, unlike $k$-anonymity, differential privacy methods have mathematical proofs of privacy guarantee. Nevertheless, applying differential privacy on graph data limits published data utility [15].

**Graph Data Publishing Under Edge-DP.** In edge-DP [50], two graphs are said to be neighboring if they differ on a single edge. Intuitively, an algorithm providing edge-DP has similar output distributions on any pair of graphs that differ in one edge, thus protecting changes to graph edges. Applying edge-DP on graph data limits its utility because graph is highly sensitive to structural changes and adding noise directly into graph data can significantly degrade its utility. To address this issue, many approaches [85; 101; 98; 97; 36] have investigated the problem of publishing anonymized graphs by indirectly perturbing graph data through a graph feature-abstraction model, such as the dK-graph model [72], hierarchical random graph (HRG) model [11], or spectral graph methods [98]. The main idea of these studies is to transform a graph into a statistical representation (e.g., degree distribution and dendrogram) or a spectral representation (e.g., adjacency matrix), then add random noise to perturb such representations, and finally generate anonymized graphs based on perturbed representations. I list some of the existing graph abstraction methods:

- *Edge-DP via dK-graph model:* The work in [85; 97] considered to approximate an original graph by $dK$-series and then perform perturbation. However, to achieve edge-DP, the global sensitivity of this approach is $O(n)$, where $n$ is the number of nodes in the original graph, thus requiring a successive amount of noise to provide differential privacy guarantees.

- *Edge-DP via HRG model:* The authors in [101] have proposed a mechanism that first projects a graph into a statistical representation (e.g., dendrogram) using a hierarchical random graph (HRG) model. Then adds random noise to perturb this representation in order to publish the entire graph by providing edge-privacy. This approach requires imposing noise proportional to global sensitivity, i.e., $O(logn)$. Later, in [36], the authors consider local sensitivity rather than global sensitivity in order to reduce noise needed to achieve differential privacy using HRG model.

- *Edge-DP via spectral graph methods:* The work in [98] proposed to project a graph to the spectral domain and inject noise to the eigenvalues and eigenvectors of the corresponding adjacency matrix. This approach requires $O(\sqrt{n})$, where $n$ is the number of nodes in the original graph.

Furthermore, some studies focused on publishing various statistical data under edge-DP, e.g., degree distribution [40], subgraph counting [51; 5], clustering coefficient [99], and frequent subgraph mining [88].

**Graph Data Publishing Under Node-DP.** In node-DP [14], two graphs are neighboring if by removing a node $v$ and all edges incident to $v$ in one graph, one obtains the other graph. Intuitively, an algorithm providing node-DP has similar output distributions on any pair of graphs that differ in one node and edges adjacent to it, thus protecting information pertaining to each individual. Node-DP is known to be more challenging to achieve and can provide stronger privacy protection than edge-DP [14; 53; 95]. Graph data is highly sensitive to structural changes under node-DP, which thus requires a large amount of noise to be added into published network statistics and can significantly degrade the utility of published network statistics.

A key technique to satisfy node-DP is "graph projection" [53; 84; 14; 16], which projects an original graph to a graph whose maximum degree is below a certain threshold $\theta$, i.e., a $\theta$-bounded graph, The motivation behind graph projection is to bound the sensitivity of publishing network statistics through a control on node degrees. I list some of the existing graph projection methods:

- *Truncation Method:* The authors of [53] have proposed a graph project technique by truncating all nodes whose degrees are larger than $\theta$ and have proven that publishing a degree histogram after truncation has a sensitivity of $2\theta S_T$, where $S_T$ is the smooth upper bound on the number of nodes whose degrees may change because of truncation. The truncation method removes more edges than necessary for the goal of ensuring a $\theta$-bounded graph.

- *Edge Removal Method:* Another graph projection technique was introduced in [5], which traverses the edges of a graph in a random order and removes all edges that are connected to a node with degree greater than $\theta$. The sensitivity for publishing subgraph counting queries (i.e., number of triangles) after this edge-removal operation has been shown to be $p(2\theta)^{p-1}$, where $p$ is the number of nodes in subgraphs.

- *Edge Addition Method:* In the work [14], projection is performed by adding edges according to *stable edge ordering*. This edge-addition approach is similar to [5], except that it inserts edges while [5] deletes edges. While this difference is minor but it is shown in [14] that the edge-addition approach preserves more edges, and publishing a degree histogram of a projected graph has a sensitivity of $2\theta + 1$.

- *Lipschitz Extension Method:* In [84] another projection method was introduced by constructing a weighted graph, and publishing degree histograms has the sensitivity of $6\theta$ in their work.

Despite considerable progress being achieved in understanding node-DP, these works have only studied the release of simple statistical data of networks (i.e., edge count [53; 95], counts of small subgraphs [5; 16], and degree distribution [53; 84; 14]). To the best of my knowledge there is no algorithm which can provide node-DP in order to publish an entire graph.

**Graph Data Publishing Under Personalized-DP (PDP).** Anonymization techniques based on personalized differential privacy (PDP) [29] where an individual can set their own privacy level $\varepsilon$ can be broadly categorised into two types:

1. *User-grained:* This approach was first presented in [32] which generalized the classic definition of differential privacy to provide freedom to each individual to have a personalised privacy budget $\varepsilon$ based on their personal privacy preferences called personalized differential privacy (PDP). Similarly, in the work [49] authors proposed mechanisms based on non-uniform sampling and personalized exponential mechanism.. In another work [62] two partitioning-based mechanisms was proposed to achieve user-grained PDP in which the dataset is partitioned to group data based on the privacy preference provided by an individual. Then, applying differential privacy mechanism on each partition using the strongest privacy protection level.

2. *Distance-grained:* These PDP approaches [56; 57] considered social networks and scale individuals' privacy based on distances between individuals in a network (e.g., length of a shortest path between two nodes).

3. *Item-grained:* These PDP approaches were first introduced for relational databases include heterogeneous differential privacy (HDP) [2], which allows different privacy levels for different data items (e.g. salary, age, etc). Later for social networks, a fine-grained approach based on distance-grained and item-grained has been presented in [108]

Despite considerable progress being achieved in understanding PDP, these works have only studied the release of single queries in a network setting [56; 57; 2; 108].

#### 2.2.4.5 Summary

Many graph data anonymization approaches have been proposed to limit protect privacy of individuals in a network setting. For clarity, Table 2.5 summarises the main idea of each graph data anonymization approach and highlights their pros and cons.

**Table 2.5**: Summary of graph data anonymization approaches.

| **Graph Data Anonymization Approaches** | | **Main Idea** | **Pros/Cons** |
|---|---|---|---|
| *k*-Anonymity based Approaches | *k*-candidate | Given a query, there exist at least *k* nodes that match the query. | Pros: Ensure identity disclosure.<br>Cons: An algorithm to construct *k*-candidate graphs was not presented. |
| | *k*-neighborhood | Organize nodes into groups according to their neighborhoods and then anonymize. the neighborhoods of nodes in the same group. | Pros: Provide guard against neighbor-graph attacks.<br>Cons: Focus only on 1-neighborhood subgraph. |
| | *k*-degree | There are at least $(k-1)$ other nodes with same degree. | Pros: Prevent degree attacks.<br>Cons: Not efficient on large graphs and degrade utility. |
| | *k*-automorphism | Divide an original graph into *k* blocks such that each subgraph corresponds to at least $(k-1)$ subgraphs. | Pros: Ensure identity disclosure.<br>Cons: Vulnerable under neighbor-graph attacks and unable to prevent membership disclosure. |
| | *k*-isomorphism | Divide an original graph into *k* disjoint subgraphs such that subgraphs in each blocks are pairwise isomorphic. | Pros: Prevent membership disclosure and provide guard against neighbor-graph attacks.<br>Cons: Unable to preserve structural similarity between original and anonymized graphs. |
| | *k*-decomposition | Divide an original graph into *k* disjoint blocks using multilevel *k*-way partitioning. | Pros: Provide guard against neighbor-graph attacks.<br>Cons: Add excessive nodes and edges into each block. |
| Graph Modification based Approaches | Random Perturbation | Modify graph by randomly adding, removing, and switching edges or nodes. | Pros: Randomness increases privacy guarantee.<br>Cons: Damage structural properties of the graph. |
| | Constraint Perturbation | Modify graph to meet some desired constraints. | Pros: Ensure identity disclosure.<br>Cons: Not efficient on large graphs. |
| Generalization or Clustering based Approaches | Vertex Clustering | Partition an original graph into disjoint sets where each super node represents at least *k* nodes. | Pros: Good source to study the macro-properties of their original graphs.<br>Cons: Graph may be shrunk after anonymization. |
| | Edge Clustering | Partition an original graph into cluster such that an edge must exists between clusters. | Pros: Preserve structure better than vertex clustering.<br>Cons: Information about local structures of an original graph is difficult to analyze. |
| | Vertex and Edge Clustering | Partitioned the original graph into clusters and then combine nodes in each cluster. | Pros: Helpful to analyze macro-properties of their original graph<br>Cons: Damage local structure of an original graph. |
| Differential Privacy | Edge-DP | Provide privacy guarantee to edges in a network. | Pros: Protect relationship information among individuals.<br>Cons: Limit data utility. |
| | Node-DP | Provide privacy guarantee to nodes in a network. | Pros: Provide stronger privacy than edge-DP.<br>Cons: More challenging to achieve than edge-DP. |
| | Personalized-DP | Allow individuals to set their own privacy preferences. | Pros: Provide personalization.<br>Cons: Difficult to achieve. |

Chapter 3

# Preliminaries

This chapter presents the basic definitions and notations that will be extensively used throughout this thesis. Table 3.1 summarises the frequently used notations.

Let $\mathcal{X}$ be a class of possible datasets. The notion of $\varepsilon$-differential privacy ($\varepsilon$-DP) [29] is defined based on the concept of neighboring datasets.

**Definition 1.** (Neighboring dataset) *Two datasets $X, Y \in \mathcal{X}$ are said to be* neighboring, *denoted as $X \sim Y$, if one can be obtained from the other by adding, deleting or modifying a single individual.*

**Definition 2.** ($\varepsilon$-Differential privacy) *A randomized mechanism $\mathcal{K}$ provides $\varepsilon$-differential privacy, if for any two neighboring datasets $X$ and $Y$, and all possible outputs $\mathcal{X}_\varepsilon \subseteq range(\mathcal{K})$, it holds*

$$Pr[\mathcal{K}(X) \in \mathcal{X}_\varepsilon] \leq e^{\varepsilon} \times Pr[\mathcal{K}(Y) \in \mathcal{X}_\varepsilon] \tag{3.1}$$

where $\varepsilon > 0$ is the differential privacy parameter. Smaller values of $\varepsilon$ provide a stronger privacy guarantee [28]. Typically, the values of $\varepsilon$ should be small, such as 0.01, 0.1, or in some cases ln 2, or ln 3 [28]. When $\varepsilon$ is very small, then $e^{\varepsilon} \approx 1 + \varepsilon$.

$\varepsilon$-differential privacy [25] was originally proposed as a privacy model to protect the responses of interactive queries to a dataset. Let $f$ be an arbitrary query function for which $\varepsilon$-differentially private response is requested. A standard way for achieving $\varepsilon$-differential privacy is by adding random noise to the true response of a query function $f$. The random noise is calibrated according to the *sensitivity* $\Delta$ of $f$. The *sensitivity* of $f$ measures the maximum variation in the query function $f$ between two neighboring datasets $X \sim Y$. Generally, there are two types of sensitivity are employed in differential privacy: *global sensitivity* and *local sensitivity*.

The global sensitivity of $f$ measures the maximum variation in the query $f$ between any two neighboring datasets $X \sim Y$ as follows.

**Definition 3.** (Global Sensitivity) *The global sensitivity of a function $f : \mathcal{X} \rightarrow \mathbb{R}^d$ is defined as:*

$$\Delta f_{GS} = \max_{X \sim Y} \|f(X) - f(Y)\|_1, \tag{3.2}$$

*where $\|.\|_1$ denotes the $L_1$-norm.*

Given a query function $f$, the amount of noise added into the response of $f$ depends on the global sensitivity and the privacy parameter $\varepsilon$, but not on the input dataset. Nissim et al. [79] proposed a local measure of sensitivity by focusing on the neighbors of an input dataset. The local sensitivity is defined as below:

**Definition 4.** (Local Sensitivity) *For a function $f : \mathcal{X} \to \mathbb{R}^d$ and a dataset $X \in \mathcal{X}$, the local sensitivity of $f$ at $X$ is*

$$\Delta f_{LS}(X) = \max_{Y} \|f(X) - f(Y)\|_1, \tag{3.3}$$

*where maximum is taken over all neighbors $Y$ of $X$.*

Note that, by Definition 3 and Definition 4, the global sensitivity $\Delta f_{GS} = max_X \;\; \Delta f_{LS}(X)$.

Generally, there are two mechanisms by which differential privacy can be achieved, i.e., *Laplace mechanism* and *Exponential mechanism*.

**Laplace Mechanism.** The mechanism was proposed by [29]. One way to satisfy differential privacy is to add noise to the output of a query $f$. In the Laplace mechanism, in order to publish $f(X)$ where $f : \mathcal{X} \to \mathbb{R}$, while satisfying $\varepsilon$-differential privacy, one publishes

$$f(X) = f(X) + Lap\left(\frac{\Delta f}{\varepsilon}\right)$$
$$\text{where} \quad \Pr[\text{Lap}(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$$

I denote a random variable drawn from a Laplace distribution with mean 0 and scale $\beta$ (equivalently variance $\sigma^2 = 2\beta^2$) as $Lap(\beta)$, which has the probability density function $Pr(x|\beta) = 1/2\beta \, exp(-|x|/\beta)$. The Laplace mechanism provides $\varepsilon$-differential privacy by adding noise from a zero-mean Laplace distribution with scale $\beta = \Delta f/\varepsilon$.

**Exponential Mechanism.** This mechanism was proposed by [75]. While the Laplace mechanism provides a solution to deal with numerical data, it cannot be applied to non-numerical data. This leads to the development of the exponential mechanism, which can be applied regardless an output of a function is numerical or not. Let $\mathcal{X}$ be the domain of input datasets, $\mathcal{S}$ be the range of noisy outputs, and $\mathbb{R}$ be the real numbers. Given a dataset $X \in \mathcal{X}$, and a quality function $q : (\mathcal{X} \times \mathcal{S}) \to \mathbb{R}$, which assigns a quality score $q(X, s)$ for outputting $s$ on input $X$, where $s \in \mathcal{S}$, the exponential mechanism outputs $s$ with a probability proportional to $exp\left(\frac{\varepsilon q(X,s)}{2\Delta_q}\right)$, where $\Delta q = max_{\forall s,X,Y}|q(X, s) - q(Y, s)|$ is the sensitivity of the quality function. This satisfies $\varepsilon$-differential privacy.

**Composition Properties.** Differential privacy satisfies several properties, including: sequential composition, parallel composition [76], and transformation invariance, i.e., post processing [54].

**Definition 5.** *(*Sequential Composition*) [76] For any sequence of mechanisms $\mathcal{K}_1, \ldots, \mathcal{K}_m$, if each $\mathcal{K}_i$ satisfies $\varepsilon_i$ differential privacy, then publishing the results of all of $\mathcal{K}_1, \ldots, \mathcal{K}_m$ satisfy $\sum_i \varepsilon_i$-differential privacy.*

**Definition 6.** *(*Parallel Composition*) [76] If $\mathcal{K}_1, \ldots, \mathcal{K}_m$ are mechanisms that access disjoint subsets of an input domain N such that each $\mathcal{K}_i$ provides $\varepsilon_i$-differential privacy then running all m mechanisms in parallel provides max($\varepsilon_i$)-differential privacy.*

**Definition 7.** *(*Post Processing*) [54] Given $\mathcal{K}_1$ that satisfies $\varepsilon$-differential privacy, and any algorithm $\mathcal{K}_2$, the new algorithm $\mathcal{K}(.) = \mathcal{K}_2(\mathcal{K}_1(.))$ satisfies $\varepsilon$-differential privacy.*

**Table 3.1**: Summary of Notations

| Notations | Description |
|---|---|
| $\mathcal{X}$ | A class of possible datasets |
| $X$ | A dataset |
| $X \sim Y$ | Two neighboring datasets |
| $r_i$ | A record in X |
| $A$ | Set of attributes |
| $\|.\|_1$ | $L_1$-norm |
| $\mathcal{K}$ | A randomized mechanism |
| $\mathcal{M}$ | A microaggregation algorithm |
| $\mathcal{M}_p$ | Partitioning function of $\mathcal{M}$ |
| $\mathcal{M}_a$ | Aggregation function of $\mathcal{M}$ |
| $\preceq_p$ | A partial ordering |
| $f$ | A query function |
| $\Delta f$ | Sensitivity of $f$ |
| $\varepsilon$ | A privacy parameter/budget/level |
| $k$ | Minimal cluster size |
| $\overline{X}$ | A microaggregated dataset |
| $X_\varepsilon$ | A differentially private dataset |
| $X \oplus Y$ | Symmetric difference between $X$ and $Y$ |
| $f \circ g$ | Composition of functions $f$ and $g$ |
| $\mathcal{G}$ | A set of all possible graphs |
| $G = (V, E)$ | A graph $G$ with the set of nodes $V$ and edges $E$ |
| $G \overset{e}{\sim} G'$ | Two edge neighboring graphs |
| $G \overset{n}{\sim} G'$ | Two node neighboring graphs |
| $deg(v)$ | Degree of node $v$ |
| $deg(G)$ | Maximum degree of nodes $v$ in $G$ |
| $f^{dK}$ | A $dK$ function |
| $\mathcal{D}$ | A set of $dK$-distributions over $G$ |
| $D$, $D_\varepsilon$ | A $dK$-distribution, a differentially private $dK$-distribution |
| $f_q$ | A degree query |
| $f_q^*$ | A microaggregated degree query |
| $t$ | A degree tuple |
| $T$ | A set of degree tuples |
| $\tau$ | A distance interval |
| $G^\theta$ | A $\theta$-bounded graph |
| $\mathcal{P}$ | A graph projection algorithm |
| $N_G(v)$ | A set of neighbors of a node $v$ in $G$ |
| $\succ_N$ | A local neighbor ordering |
| $\succ_V$ | A global node ordering |
| $\succ_\Gamma$ | A total ordering on edges in $E$ |
| $\Gamma$ | A stable ordering |
| $\Phi$, $\Phi^v$ | Privacy specification, privacy specification of node $v$ |
| $\mu$ | A global threshold |

# Part I

# Relational Data Publishing

Chapter 4

# 2-Stable Microaggregation: Publishing Relational Datasets with Differential Privacy

## 4.1 Overview

In this chapter, I study the problem of publishing relational data under the guarantee of differential privacy while enhancing relational data utility. I observe that microaggregation [18] can be integrated into differential privacy for generating utility-preserving differentially private datasets for relational data. Particularly, the amount of noise needed to achieve differential privacy can be greatly reduced if the query is run on a dataset obtained through microaggregation, i.e., microaggregated dataset. In doing so, the original query $f$ is approximated by $f \circ \mathcal{M}$, since $f$ is run on the microaggregated dataset rather than the original dataset, where $\mathcal{M}$ is a microaggregation algorithm. However, an arbitrary microaggregation algorithm cannot reduce sensitivity when being incorporated into differential privacy as shown in literature [90; 91] and further explained in Example 1. To address this issue, the notion of *insensitive microaggregation*[90; 91] was proposed. It uses microaggregation to achieve $k$-anonymity in which a certain correspondence between clusters in the microaggregated datasets of two neighboring datasets is imposed. Therefore, the noise added to guarantee differential privacy can be greatly reduced. However, insensitive microaggregation still has the limitations: (1) it yields worse within-cluster homogeneity due to a total order relation required for the distance function [91], and (2) the minimum cluster size $k$ grows with the size $n$ of the dataset and one thus needs $k \geq \sqrt{n}$ to reduce noise.

To alleviate these issues, I have proposed a microaggregation-based framework for generating $\varepsilon$-differentially private datasets for relational data based on a notion of *2-stable microaggregation*. The proposed framework provides better within cluster homogeneity by ensuring that at most two pairs of corresponding clusters in microaggregated neighboring datasets differ in a single record as compared to insensitive microaggregation, where at most all pairs of corresponding clusters differ in a single record. Thus, when $k \geq 2$, the addition of noise can always be reduced in comparison with directly applying differential privacy over an original dataset, regardless of the size of a dataset.

The main contributions of this chapter are as follows:

- I present a microaggregation-based framework for generating $\varepsilon$-differentially private datasets and formulate a novel notion of *2-stable microaggregation* to characterize the correspondence of clusters in microaggregated neighboring datasets.
- I propose a *2-stable microaggregation algorithm* that can ensure the correspondence of clusters in the microaggregated datasets of two neighboring datasets.
- I experimentally verify the utility reduction of our proposed framework on two real-world datasets containing numerical data. It shows that our algorithm can effectively enhance the utility of released data by providing better within-cluster homogeneity and reducing the amount of noise, in comparison with the state-of-the-art methods.

The rest of this chapter is organized as follows. Section 4.2 introduces the problem definition. Section 4.3 presents the framework *2-stable microaggregation* for generating $\varepsilon$-differentially private datasets. Section 4.4 discusses algorithms for microaggregating reltional datasets. Section 4.5 discusses the experimental results, which empirically verify the noise reduction and privacy guarantee of the proposed algorithm against the baseline algorithms. Section 4.6 summarises the chapter.

## 4.2 Problem Definition

Let $\mathcal{X}$ be a class of possible datasets. A relational dataset $X \in \mathcal{X}$ consists of a set of records, each $r_i \in X$ being associated with a set of attributes $A$. We assume that each individual has only one record in a dataset $X$.

**Definition 8.** *(*Neighboring datasets*) Two datasets $X, Y \in \mathcal{X}$ are said to be* neighboring, *denoted as $X \sim Y$, if $|X| = |Y| = n$, but $X$ and $Y$ differ in one record.*

Given a dataset $X$, the goal is to generate $X_\varepsilon$ (an anonymized version of $X$) that can provide $\varepsilon$-differential privacy guarantee for protecting the privacy of individuals' records in $X$. Below, the notion of $\varepsilon$-differentially private datasets is formulated.

**Definition 9.** *(*Differentially private datasets*) A randomized mechanism $\mathcal{K} : \mathcal{X} \to \mathcal{X}$ provides $\varepsilon$-differentially private datasets, if for each pair of neighboring datasets $X \sim Y$, and all possible outputs $\mathcal{X}_\varepsilon \subseteq range(\mathcal{K})$, it holds*

$$Pr[\mathcal{K}(X) \in \mathcal{X}_\varepsilon] \leq e^\varepsilon \times Pr[\mathcal{K}(Y) \in \mathcal{X}_\varepsilon] \tag{4.1}$$

where $\varepsilon > 0$ is the differential privacy parameter. Smaller values of $\varepsilon$ provide stronger privacy guarantees [28].

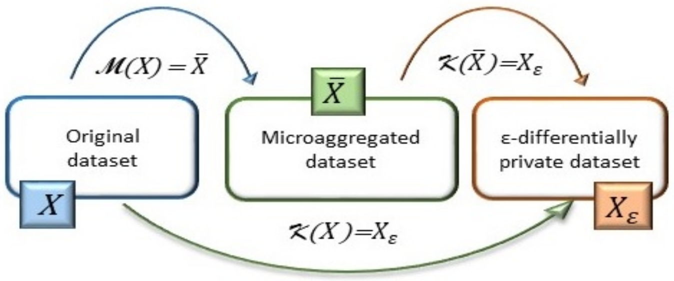

$\mathcal{M}(X) = \overline{X}$
$\mathcal{K}(\overline{X}) = X_\varepsilon$
Original dataset
$X$
$\overline{X}$
Microaggregated dataset
ε-differentially private dataset
$X_\varepsilon$
$\mathcal{K}(X) = X_\varepsilon$

**Figure 4.1**: Problem setting.

Let $f$ be a query function that extracts data against records in the dataset. Following the previous works [89; 70], differentially private datasets can be generated by considering data publishing as the answers to subsequent queries for each record in the dataset. For numerical data, the addition of noise can be drawn from a Laplace distribution by first computing the answer $f(X)$ and then generating the noisy answer $f(X) = f(X) + Lap(\Delta(f)/\varepsilon)$ to provide $\varepsilon$-differential privacy, where $\Delta(f)$ is the *global sensitivity* $\Delta$ of $f$.

In this chapter, the aim is to generate $\varepsilon$-differentially private datasets for relational data by using microaggregation for improving data utility. Given a dataset $X$, a microaggregated dataset $\overline{X}$ is created by a microaggregation algorithm $\mathcal{M}$ in two stages:

- First, $X$ is partitioned into a set of clusters $\mathcal{C}_X$, such that each cluster in $\mathcal{C}_X$ has at least $k$ records, where $k$ is a preset constant value, and the records within each cluster are as similar as possible (homogeneous).

- Second, it aggregates each cluster in $\mathcal{C}_X$ by replacing each record with the representative (average) record of the cluster.

As illustrated in Figure 4.1, a microaggregated dataset $\overline{X}$ resulting from running $\mathcal{M}$ over $X$ is added between $X$ and $X_\varepsilon$ to increase utility of $X_\varepsilon$. In doing so, the original query $f$ is approximated by $f \circ \mathcal{M}$, since $f$ is run on the microaggregated dataset $\overline{X}$ rather than the original dataset $X$. This thus introduces two kinds of errors: one is the random noise, which depends on the sensitivity $\Delta(f)$ of query $f$ to guarantee $\varepsilon$-differential privacy, and the other one is due to computing $f$ over $\overline{X}$ instead of $X$. As will be discussed in Section 4.5, the first kind of error is much larger than the second kind of error in terms of the information loss in $\varepsilon$-differentially private datasets. To increase the overall utility, the key challenge is how to reduce $\Delta(f \circ \mathcal{M})$ such that $\Delta(f \circ \mathcal{M}) \leq \Delta(f)$.

## 4.3 2-Stable Microaggregation Framework

In this section, the microaggregation-based framework for generating differentially private datasets for relational data, called 2-*stable microaggregation*, is presented.

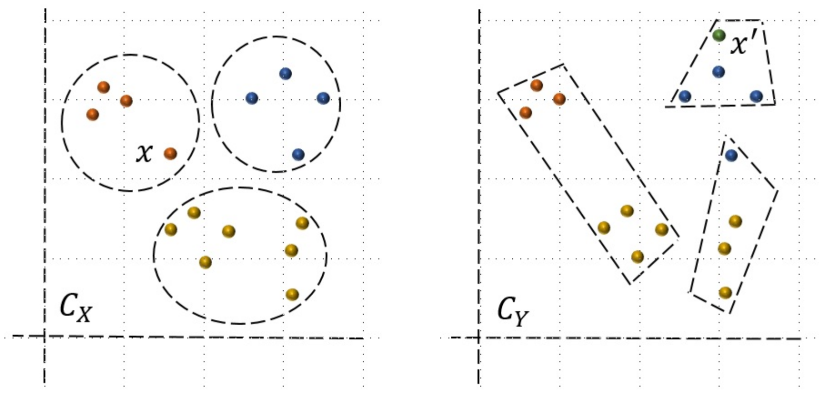


**Figure 4.2**: Clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ generated by $\mathcal{M}$ over $X \sim Y$.

Given two neighboring datasets $X \sim Y$ that only differ in a single record, their microaggregated datasets $\overline{X}$ and $\overline{Y}$ by an arbitrary microaggregation algorithm $\mathcal{M}$ may generate considerably different clusters.

**Example 1.** *For instance, suppose that a record $x$ in $X$ in Figure 4.2 is modified to $x'$ in $Y$, i.e., $X \sim Y$. A microaggregation algorithm $\mathcal{M}$ (e.g. MDAV [21]) with $k = 4$ can generate $\mathcal{C}_X$ and $\mathcal{C}_Y$ over $X$ and $Y$, respectively. Although $X$ and $Y$ only differ in one record, the clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ are completely unrelated.*

In this case, the maximum variation between one cluster from $C_X$ and another unrelated cluster from $C_Y$ is $\Delta(f)$. Since there are $n/k$ clusters in $C_X$ and $C_Y$, $\Delta(f \circ \mathcal{M}) = n/k \times \Delta(f)$, which can be significantly higher than $\Delta(f)$ when the datasets are large, i.e., $n$ is large. This leads to a much larger $\Delta(f \circ \mathcal{M})$ than $\Delta(f)$. Thus, an arbitrary $\mathcal{M}$ could not reduce sensitivity when incorporated into differential privacy.

To address the above issue, the notion of *insensitive* microaggregation was proposed [91].

**Definition 10.** *(*Insensitive Microaggregation *[91]) A microaggregation algorithm $\mathcal{M}$ is said to be* insensitive *if, for every pair of neighboring datasets $X \sim Y$, there is a bijection between the set of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ resulting from running $\mathcal{M}$ on $X$, and $Y$, respectively, such that each pair of corresponding clusters differs at most in one single record.*

This implies that the maximum variation between each pair of corresponding clusters is reduced to $1/k \times \Delta(f)$. Since there are still $n/k$ clusters, $\Delta(f \circ \mathcal{M})$ is $n/k \times \Delta(f)/k$. As a result, insensitive microaggregation may greatly reduce sensitivity as compared with $n/k \times \Delta(f)$ for standard microaggregation (e.g. MDAV [21]). However, insensitive microaggregation still has some limitations.

- First, to achieve $\Delta(f \circ \mathcal{M}) \leq \Delta(f)$ as desired, $(n/k \times \Delta(f)/k) \leq \Delta(f)$ must hold. Therefore, one needs $k \geq \sqrt{n}$ in order to reduce added noise in comparison with directly applying $\mathcal{K}$ over $X$ [89]. For large datasets, $k$ thus needs to be large enough for reduced sensitivity.

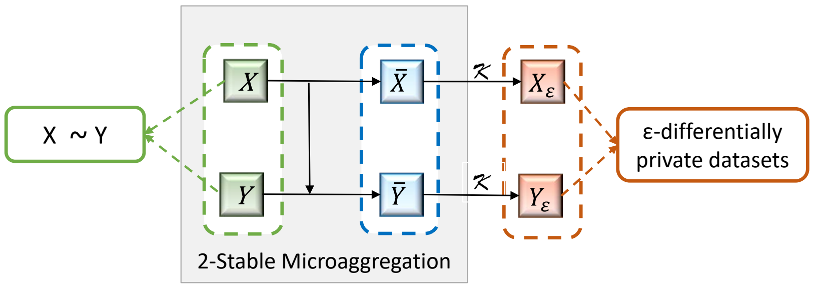


**Figure 4.3**: A high-level overview of 2-*Stable Microaggregation*.

- Second, as noted in the work [89] and will also be discussed in Section 4.5, the clusters generated by insensitive microaggregation are often less homogeneous than the clusters generated by standard microaggregation, such as MDAV [21]. This is because, to ensure the insensitive property, the distance function used by insensitive microaggregation algorithms must be consistent with the total order relation $\leq_X$ [91].

To alleviate these limitations, I define the notion of 2-*stable microaggregation*.

**Definition 11.** (2-Stable microaggregation) *Let $\mathcal{M}$ be a microaggregation algorithm, $\mathcal{C}_X = \{c_1, ..., c_n\}$ be the set of clusters that results from running $\mathcal{M}$ on $X$, and $\mathcal{C}_Y = \{c'_1, ..., c'_n\}$ be the set of clusters that results from running $\mathcal{M}$ on $Y$. $\mathcal{M}$ is* 2-stable *if, for every pair of neighboring datasets $X \sim Y$, there is a bijection between $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that at most two pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ differ in a single record.*

Since stable microaggregation affects at most two pairs of corresponding clusters in $C_X$ and $C_Y$, $\Delta(f \circ \mathcal{M})$ is further reduced to $(2 \times \Delta(f)/k)$ as compared to $(n/k \times \Delta(f)/k)$ for insensitive microaggregation. Thus, when $k \geq 2$, the addition of noise can always be reduced in comparison with directly applying $\mathcal{K}$ over $X$, regardless of the size of a dataset.

A high-level description of the proposed framework is presented in Figure 4.3, in which 2-stable microaggregation is applied to generate $\overline{X}$ and $\overline{Y}$ over $X \sim Y$. Then $\varepsilon$-differentially private datasets $X_\varepsilon$ and $Y_\varepsilon$ are generated by applying $\mathcal{K}$ over $\overline{X}$ and $\overline{Y}$, respectively.

## 4.4 Proposed 2-Stable Microaggregation Algorithm

In this section, we present algorithm for microaggregating datasets under 2-*stable microaggregation* framework.

### 4.4.1 S-MDAV Algorithm

To achieve 2-stable property, we propose an algorithm which takes a pair of neighboring datasets $X \sim Y$ and generates cluster $\mathcal{C}_X$ with a standard microaggregation

**Algorithm 1:** *S-MDAV Algorithm*

```
   Input: $X \sim Y$ where $r := X - Y$ and $r' := Y - X$
          $\mathcal{M}$ : a standard microaggregation algorithm
   Output: $\overline{X}, \overline{Y}$
1  $\mathcal{C}_X \leftarrow \{c_1, ..., c_n\}$ generated by $\mathcal{M}_p$ over $X$
2  $\mathcal{C}_Y \leftarrow replace(\mathcal{C}_X, r, r')$
3  $D, L := \phi$
4  foreach $c_i \in \mathcal{C}_X$ do
5      $D := D \cup \{(dist(r', r_{c_i}), c_i)\}$
6  $d_{min}, c_{min} \leftarrow \mathtt{Min}(D)$
7  if $r' \in c_{min}$ then
8      $\overline{Y} \leftarrow \mathcal{M}_a(\mathcal{C}_Y)$
9  else
10     $c_i := c(r')$
11     $D := D - \{(\mathcal{G}_{dist}(D, c_i), c_i)\}$
12     foreach $c_j \in \mathcal{C}_X \setminus \{c_i\}$ do
13         $d_j \leftarrow \mathcal{G}_{dist}(D, c_j)$
14         $D := D - \{(d_j, c_j)\} \cup \{(dist(r_{c_i}, r_{c_j}) + d_j, c_j)\}$
15     $d_{min}, c_{min} \leftarrow \mathtt{Min}(D)$
16     foreach $r_i \in c_{min}$ do
17         $swap(\mathcal{C}_Y, r', r_i)$
18         $\overline{Y}_i \leftarrow \mathcal{M}_a(\mathcal{C}_Y)$
19         $L := L \cup \{(\mathcal{I}_{loss}(Y_i, \overline{Y}_i), \overline{Y}_i)\}$
20     $\overline{Y} \leftarrow \mathtt{Min}(L)$
21 $\overline{X} \leftarrow \mathcal{M}_a(\mathcal{C}_X)$
22 Return $\overline{X}, \overline{Y}$
```

algorithm (e.g., MDAV [21]). Then based on $\mathcal{C}_X$ we generate $\mathcal{C}_Y$ such that clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ are 2-stable yet homogeneous.

Algorithm 1 describes the details of our proposed S-MDAV algorithm. Given $X \sim Y$, I start with partitioning the dataset $X$ into $\mathcal{C}_X$ by $\mathcal{M}_p$, i.e., the partition function of a microaggregation algorithm $\mathcal{M}$. Then I replace the record $r \in X$ with $r'$ and initialize two empty lists $D$ and $L$ to store distances and information loss values (Lines 1-3). For each cluster $c_i \in \mathcal{C}_X$, by means of function $dist()$, I compute distance between $r'$ and $r_{c_i}$, where $r_{c_i}$ is the representative record of $c_i$ to compute and store the distances between $r'$ and other cluster representatives. Then, I compute $d_{min}$, i.e., the minimum distance in $D$, and $c_{min}$, i.e., the cluster in $\mathcal{C}_X$ with $d_{min}$, by means of `Min` function (Lines 4-6). If $r'$ is in the cluster $c_{min}$ of $\mathcal{C}_Y$, then I aggregate $\mathcal{C}_Y$ by $\mathcal{M}_a$ that is the aggregation function of the microaggregation algorithm $\mathcal{M}$ (Lines 7-8). In this case, only one pair of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ is affected as homogeneity cannot be enhanced further. Otherwise, for each cluster $c_j \in \mathcal{C}_X \setminus \{c_i\}$ where $r' \in c_i$, I compute the distance between the representative records of clusters $c_i$ and $c_j$, i.e.,

$r_{c_i}$ and $r_{c_j}$ to find records for swapping. I proceed with updating $D$ by summing up both distances of the corresponding clusters, excluding the distance of $c_i$ obtained by function $\mathcal{G}_{dist}$ (Lines 10-14). In order to get the cluster $c_j$ that is of the minimum distance from $c_i$, I find $d_{min}$, i.e., the minimum distance, and $c_{min}$, i.e., the cluster $c_j \in \mathcal{C}_X$ with $d_{min}$ from $D$, by means of `Min` function. After that, I swap $r'$ in $c_i$ of $\mathcal{C}_Y$ with each record $r_i$ in $c_{min}$ of $\mathcal{C}_Y$, and compute $\mathcal{C}_Y$ with the minimum information loss by function $\mathcal{I}_{loss}$ (Lines 15-20). In this case, at most two pairs of clusters differ at most in a single record. The algorithm terminates by returning the microaggregated datasets $\overline{X}$ and $\overline{Y}$ that have the minimum information loss.

**MDAV Vs S-MDAV Algorithm.** A microaggregation algorithm MDAV (maximum distance to average vector) [21] is a fixed-size microaggregation algorithm, in which all clusters have exactly $k$ records, except the last one, which has between $k$ and $2k$-1 records. As our proposed algorithm S-MDAV use MDAV, we recall the MDAV algorithm which generates clusters as follow:

(i) Compute the centroid $c$ of records in the dataset. Find the most distant record $r$ from the centroid. Also find the most distant record $s$ from $r$.

(ii) Form a cluster with $r$ and the $k$-1 records closest to $r$; form a cluster with $s$ and the $k - 1$ records closest to $s$.

(iii) If there are at least $2k$ records which do not belong to any of the clusters formed in Step 2. Repeat Step 1 by minus the clusters formed in Step 2.

(iv) If there are between $k$ and $2k$-1 records which do not belong to any of the clusters formed in Step 2, form a new cluster and exit the algorithm.

(v) If there are less than $k$ records which do not belong belong to any of the clusters formed in Step 2, add them to the closest cluster whose centroid is closest to the centroid of remaining records.

A standard microaggregation algorithm MDAV generate homogeneous cluster, however, cannot reduce sensitivity when being incorporated into differential privacy as shown in [91] and in Example 1. The proposed S-MDAV algorithm uses the standard microaggregation algorithm MDAV to generate the clusters in $\mathcal{C}_X$ as well as most of the clusters in $\mathcal{C}_Y$ and then achieve 2-stable property by swapping records. In doing so, S-MDAV decreases the sensitivity of $f \circ \mathcal{M}$ and thus reduces the errors caused by microaggregation.

## 4.5 Experiments

I have evaluated the proposed framework to study how 2-stable microaggregation enhances the utility of differentially private datasets.

### 4.5.1 Experimental Setup

**Datasets.** I used two datasets in the experiments:

1. CENSUS dataset[1] contains 1,080 records [91; 21; 89]. As in [91] I took 4 numerical attributes FEDTAX (Federal income tax liability), FICA (Social security retirement payroll deduction), INTVAL (Amount of interest income) and POTHVAL (Total other persons income).

2. EIA dataset[1] contains 4,092 records [18]. I took 4 numerical attributes attributes RESREVENUE (Revenue from sales to residential consumers), RESSALES (Sales to residential consumers), TOTREVENUE (Revenue from sales to all consumers), and TOTSALES (sales to all consumers).

Following [91], I considered the sensitivity of an attribute to be the difference between the lower bound (i.e. 0) and upper bound (1.5 $\times$ the maximum value) of the attribute. For both CENSUS and EIA datasets, the value of $k$ is set to between 2 and 100.

**Evaluation measure.** I used the measure $IL1s$ [109] to compute the *information loss* between the original and differentially private datasets. Formally, for each record $r_i$,

$$IL1s = \frac{1}{|A| \cdot n} \sum_{i=1}^{n} \sum_{j=0}^{|A|} \frac{|x_{ij} - x'_{ij}|}{\sqrt{2}S_j} \tag{4.2}$$

where $|A|$ is the number of attributes, $n$ is the number of records in the dataset, $x_{ij}$ is the value of attribute $a_j \in A$ for record $r_i$ in the original dataset, $x'_{ij}$ is the value of attribute $a_j \in A$ for record $r_i$ in the corresponding differentially private dataset, and $S_j$ is the standard deviation of attribute $a_j \in A$ in the original dataset.

**Baseline Methods.** I considered the following baseline methods:

1. MDAV, which is a standard microaggregation algorithm [21].

2. I-MDAV, which is an insensitive microaggregation algorithm proposed in [91].

3. $\varepsilon$-DP, which is a standard $\varepsilon$-differential privacy algorithm in which noise is added using the Laplace mechanism [29].

4. S-MDAV is the proposed 2-stable microaggregation algorithm, which extends MDAV in partitioning and aggregation.

### 4.5.2 Experimental Results and Discussion

I first conducted experiments to compare the information loss of microaggregated datasets that are generated by MDAV and I-MDAV under varying $k$ between 2 to

[1]http://neon.vb.cbs.nl/casc/CASCtestsets.htm

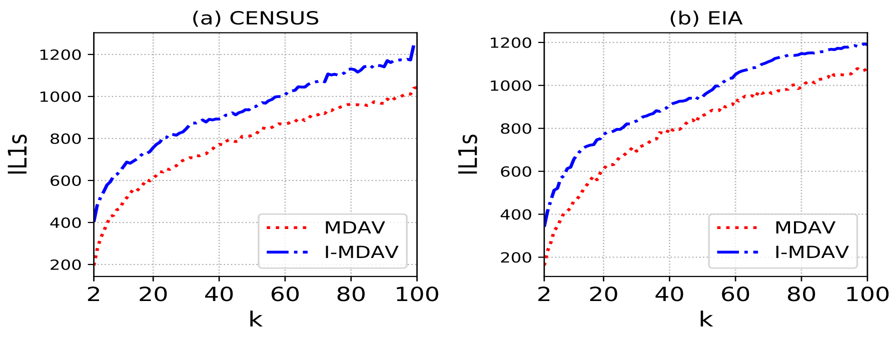


**Figure 4.4:** Evolution of *IL1s* using MDAV and I-MDAV for different values of $k$: (a) CENSUS and (b) EIA.

100. The results are shown in Figure 4.4. It can be observed that, for both CENSUS and EIA datasets, the information loss of microaggregated datasets is less with MDAV as compared to I-MDAV. This is because the clusters generated by MDAV are more homogeneous than the clusters generated by I-MDAV. As I used MDAV in our algorithm S-MDAV to generate the clusters in $C_X$ as well as most of the clusters in $C_Y$, S-MDAV decreases the sensitivity of $f \circ \mathcal{M}$ and thus reduces the errors caused by microaggregation.

Then, to verify the overall utility of $\varepsilon$-differentially private datasets, I conducted experiments to compare the information loss between the original and $\varepsilon$-differentially private datasets generated by using our algorithm S-MDAV and the baseline methods I-MDAV and $\varepsilon$-DP. Figures 4.5 and 4.6 present our experimental results. For $\varepsilon$-DP, I used the following privacy parameters $\varepsilon = [0.01, 0.1, 1.0, 10.0]$, which cover the range of differential privacy levels widely used in the literature [28; 70; 89]. For each parameter setting of $\varepsilon$, I ran 3 times and take the average result. The information loss for $\varepsilon$-DP is displayed as horizontal lines, as $\varepsilon$-DP does not depend on $k$.

Regarding the evolution of *IL1s* values shown in Figures 4.5 and 4.6, it can be seen that, for every value of $\varepsilon$, I-MDAV is only able to achieve $\Delta(f \circ \mathcal{M}) \leq \Delta(f)$ if $k \geq \sqrt{n}$, i.e., ($k = \sqrt{1,080} \approx 33$ for CENSUS and $k = \sqrt{4,092} \approx 64$ for EIA). This is consistent with the previous discussion in Section 4.3. Nonetheless, this also means that for large datasets I-MDAV requires $k$ to be enough large in order to effectively reduce $\Delta(f \circ \mathcal{M})$, i.e., the size of $k$ grows with the size of a dataset $n$. In contrast, for S-MDAV, as stated in Section 4.3, one needs $k \geq 2$ to reduce $\Delta(f \circ \mathcal{M})$ as compared to $\varepsilon$-DP. As the experiments show that our proposed algorithm S-MDAV leads to less information loss for every value of $\varepsilon$ as compared to I-MDAV and $\varepsilon$-DP in both CENSUS and EIA datasets. This is because the sensitivity $\Delta(f \circ \mathcal{M})$ is significantly reduced when S-MDAV is used for microaggregation.

I have also noticed that by approximating a query $f$ to $f \circ \mathcal{M}$ via microaggregation, the errors caused by random noise that depends on the sensitivity of $f \circ \mathcal{M}$ dominate the impact on the utility of differentially private datasets generated via microaggregation, compared to the errors existing between the original and microaggregated datasets.

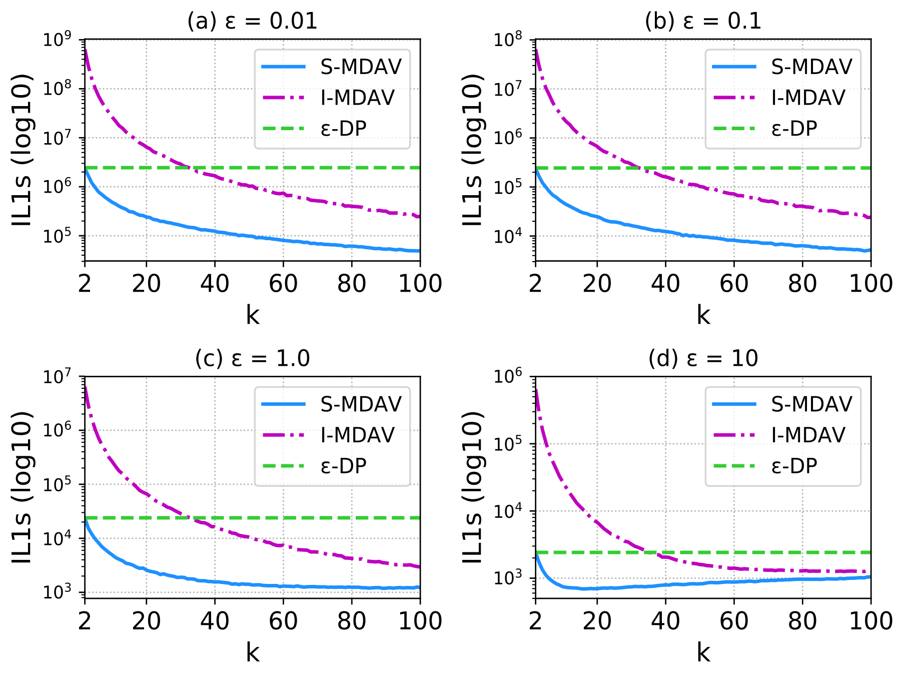


**Figure 4.5:** Evolution of *IL1s* using S-MDAV, I-MDAV and $\varepsilon$-DP for different values of $k$ and $\varepsilon$ in CENSUS.

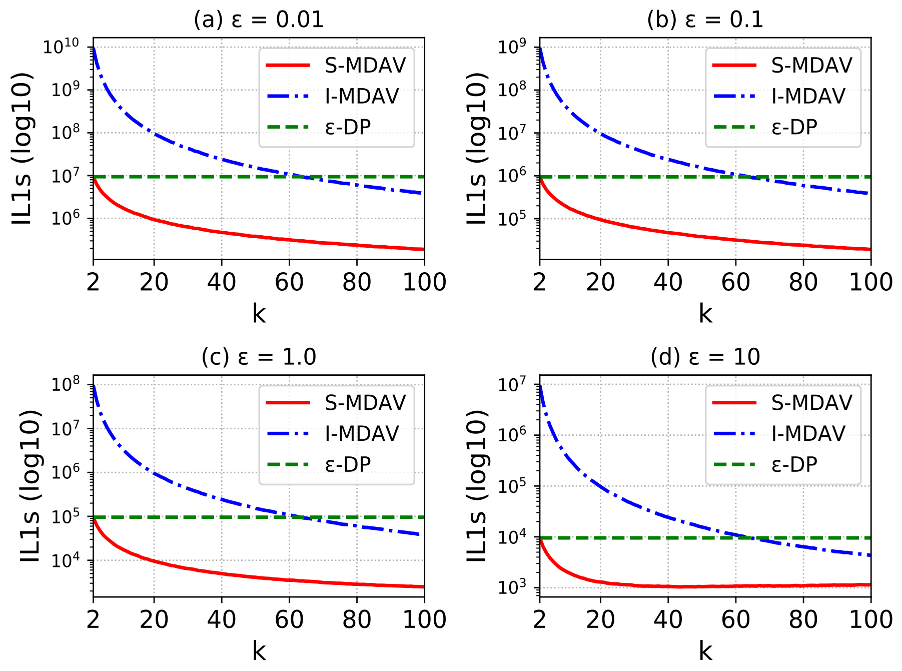


**Figure 4.6:** Evolution of *IL1s* using S-MDAV, I-MDAV and $\varepsilon$-DP for different values of $k$ and $\varepsilon$ in EIA.

On the other hand, there is a trade-off between level of homogeneity because of the constraint imposed by 2-stable microaggregation that at most two pairs of corresponding clusters can differ in a single record. As during microaggregation process the level of homogeneity can only be achieved up to some extend which limits 2-stable microaggregation algorithm to achieve local optimal within cluster homogeneity restricting the error reduction during microaggregation process. However, this constraint imposed by 2-stable microaggregation helps to reduce the overall sensitivity added into published data to achieve differential privacy, thus enhancing the overall utility of differentially private datasets. In order to address this issue I extend this preliminary work and propose a *general* framework in the next chapter.

## 4.6 Summary

In this chapter, I have introduced a microaggregation-based framework that incorporates microaggregation and differential privacy into the data publishing process. I formulated a new notion of *2-stable microaggregation* to characterize a desired property of microaggregation and further developed a simple yet effective *2-stable microaggregation algorithm*. The utility enhancement of the proposed framework has been empirically verified over two real-world datasets. The experiments demonstrated that the proposed framework outperforms the state-of-the-art methods by reducing the noise added into differentially private datasets significantly, regardless of the size of a dataset.

Chapter 5

# $\alpha$-Stable Microaggregation: Publishing Relational Datasets with Differential Privacy

## 5.1 Overview

In this chapter, I extend the the preliminary work described in Chapter 4 and propose a *general* framework. Although the amount of noise added into differentially private datasets can be considerably reduced, regardless of the size of a dataset, by using 2-stable microaggregation, however, 2-stable microaggregation still has the limitations:

- Restricting the cluster correspondence in microaggregated datasets of any two neighboring datasets such that at most two pairs of corresponding clusters differ in a single record can limit an algorithm to achieve local optimal within cluster homogeneity. This thus leads to limiting the error reduction during microaggregation process which can affect the utility of generated differentially private datasets.
- 2-stable microaggregation is unable to achieve the insensitive property due to the lack of ordering over elements (records) in $X \sim Y$. Thus, it does not guarantee differential privacy if we go beyond a specific pair of neighboring datasets. However, differential privacy is designed for any two neighboring datasets $X$ and $Y$. This thus affects the privacy guarantee of generated differentially private datasets.

To alleviate limitations of 2-stable microaggregation, I propose a general framework called *$\alpha$-stable microaggregation* which generalizes *2-stable microaggregation* [46]. The general framework introduces a parameter $\alpha$, where 2-stable microaggregation becomes a special case of this general framework with $\alpha = 2$, and insensitive microaggregation becomes a special case of this general framework with $\alpha = n/k$. $\alpha$-stable microaggregation enforces that at most $\alpha$ pairs of corresponding clusters in microaggregated neighboring datasets can differ in a single record to enhance within cluster homogeneity by requiring a partial ordering $\preceq_p$ over the elements in $X \sim Y$. Thus, the amount of noise added to differentially private datasets can always be controlled by the parameter $\alpha$. Conceptually, $\alpha$ indicates the trade-off between errors introduced during microaggregation and differential privacy.

The main contributions of this chapter are as follows:

- I present a general microaggregation-based framework for generating $\varepsilon$-differentially private datasets and formulate a notion of *α-stable microaggregation* to characterize the correspondence of clusters in microaggregated datasets.
- I propose two algorithms under the $\alpha$-stable microaggregation framework which are based on a partial order of elements (records) $\preceq_p$ in a dataset that can ensure the correspondence of clusters in the microaggregated datasets of any two neighboring datasets.
- I experimentally verify the utility and privacy trade-off in microaggregated and differentially private datasets achieved by the proposed framework on three real-world datasets containing numerical data. It shows that the algorithms can effectively enhance the utility of released data by providing better within-cluster homogeneity and reducing the amount of noise added into differentially private datasets significantly.

The rest of this chapter is organized as follows. Section 5.2 defines the problem of generating utility preserving differential private datasets. Section 5.3 presents our proposed framework based on microaggregation to generate differentially private datasets. Section 5.4 presents the proposed $\alpha$-stable algorithms. Section 5.5 introduces the partial order property. Section 5.6 discuss the experimental design and results. Section 5.8 summarises the chapter.

## 5.2 Problem Definition

To generate differentially private datasets, we define the notion of *identity function* to formulate queries on $X$.

**Definition 12.** (Identity function*) An* identity function $f_{r_i}()$ *returns the attribute values corresponding to the i-th record of X, for i between* 1 *and n (the number of records in X).*

Conceptually, $f_{r_i}()$ returns data against a specific record in $X$. Assume that we have a dataset $X$ with $n$ records and we want to generate $X_\varepsilon$. Let $f_{r_i}()$ be an identity function that returns data against a specific record in $X$, then to generate $X_\varepsilon$ under the guarantees of differential privacy, we query $X$ with $f_{r_i}(X)$, for $i = 1$ to $n$, and perform perturbation by adding controlled noise to the true response of each $f_{r_i}(X)$ through Laplace mechanism [29]. Let $\mathcal{K}_{r_i}$ be a randomize mechanism for perturbation and $\mathcal{K}_{r_i}(f_{r_i}(X))$ perturbs each response returned by the identity functions. Thus, we can view $\mathcal{K}_{r_i}$ over each identity function $f_{r_i}(X)$ for $i = 1$ to $n$ perturbs the entire dataset $X$. To release a dataset under the guarantees of differential privacy, I perturb a dataset $X$ by adding controlled noise from Laplace mechanism [29] defined in Chapter 3. Formally, I define the following Laplace mechanism to perturb each response returned by the identity functions over a dataset $X$.

**Definition 13.** *(*Perturbation*) Let $\varepsilon > 0$ be the* privacy parameter *(smaller values provide stronger privacy guarantees). The following Laplace mechanism is applied to produce a perturbed output of $f_{r_i}(X)$ for i between* 1 *to n:*

$$\mathcal{K}_{r_i}(f_{r_i}(X)) = f_{r_i}(X) + Lap\left(\frac{\Delta f_{r_i}}{\varepsilon}\right)^d$$
$$where \quad \Delta f_{r_i} = \max_{X \sim Y} \|f_{r_i}(X) - f_{r_i}(Y)\|_1$$
$$and \quad Pr[Lap(\beta) = x] = \frac{1}{2\beta} e^{-|x|/\beta}$$

$\Delta f_{r_i}$ refers to the *sensitivity* of of each individual query $f_{r_i}()$. While satisfying differential privacy, *sensitivity* determines how much perturbation is required to release response of query $f_{r_i}()$. If we query $X$ with a set of identity functions $f = \cup_{i=1}^{n} \{f_{r_i}\}_{r_i \in X}$ and the response to each $f_{r_i}$ perturbs by $\mathcal{K}_{r_i}$ satisfies $\varepsilon$-differential privacy, as each query refers to a different record, by the parallel composition property of differential privacy [76], we can generate $X_\varepsilon$ that satisfies $\varepsilon$-differential privacy.

However, the total amount of random noise being added into the responses of identity functions to achieve $\varepsilon$-differential privacy can be very high, as each identity function $f_{r_i}()$ refers to a single record in $X$, the sensitivity $\Delta$ of $f_{r_i}()$ is large (the diameter of the domains of records in $X$). This thus leads to degrade the utility of published $\varepsilon$-differentially private dataset $X_\varepsilon$. To control the amount of random noise and thus increase the utility of $X_\varepsilon$ we microaggregate similar records in $X$ before adding noise.

Given a dataset $X$, a *microaggregation* algorithm $\mathcal{M}$ can generate a microaggregated dataset $\overline{X}$, i.e., $\mathcal{M}(X) = \overline{X}$ in two stages. First, $X$ is partitioned into a set of clusters $\mathcal{C}_X$, such that each cluster in $\mathcal{C}_X$ has at least $k$ records, where $k$ is a preset constant value, and the records within each cluster are homogeneous (similar). Second, $\mathcal{M}$ aggregates each cluster in $\mathcal{C}_X$ by replacing each record with the representative record (centroid) of the cluster it belongs to. Then the $\varepsilon$-differentially private dataset $X_\varepsilon$ is generated from microaggregated dataset $\overline{X}$ by taking collection of $\varepsilon$-differentially private responses to identity functions $f_{r_i}(\overline{X})$, i.e., $f = \cup_{i=1}^{n} \{f_{r_i}\}_{r_i \in \overline{X}}$. Note that both $X$ and $\overline{X}$ have $n$ records and the *i-th* record in $\overline{X}$ refers to the *i-th* record in $X$.

Due to the prior step of microaggregation $\mathcal{M}$, the sensitivity of queries $f(\overline{X})$ used to generate $\varepsilon$-differentially private dataset $X_\varepsilon$, reflects the effect that modification suffered by a single record in $X$ is distributed among groups of multiple records in $\overline{X}$. Thus, we start thinking in terms of groups of $k$ records instead of individual record. As each record in $\overline{X}$ depends on $k$ records in $X$, this leads to reduce sensitivity of the representative records (centroids) as compared to the sensitivity of the original records, hence improving data utility by adding less noise into differentially private datasets.

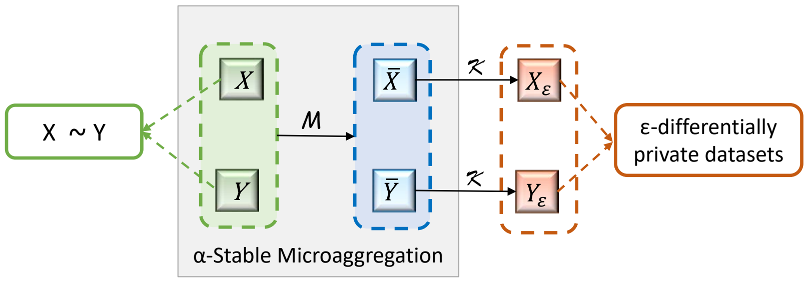


**Figure 5.1**: A high-level overview of $\alpha$-Stable Microaggregation.

Our goal is to design a general microaggregation framework and develop microaggregation algorithms under the proposed framework to anonymizing datasets under differential privacy.

## 5.3 $\alpha$-Stable Microaggregation Framework

In this section, a general framework for generating differentially private datasets, based on a notion of microaggregation, called *$\alpha$-stable microaggregation* is presented.

**Definition 14.** ($\alpha$-Stable microaggregation) *Let $\mathcal{M}$ be a microaggregation algorithm, $\mathcal{C}_X = \{c_1, ..., c_m\}$ be the set of clusters resulting from running $\mathcal{M}$ on X, and $\mathcal{C}_Y = \{c'_1, ..., c'_m\}$ be the set of clusters resulting from running $\mathcal{M}$ on Y. $\mathcal{M}$ is* $\alpha$-stable *if, for every pair of neighboring datasets $X \sim Y$, there is a bijection between $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that at most $\alpha$ pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ differ in a single record, where $\alpha \in [1, m]$ is a user defined parameter.*

For convenience of expression, we also say that, if there is a bijection between $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that at most $\alpha$ pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ differ in a single record, $\mathcal{C}_X$ and $\mathcal{C}_Y$ are *$\alpha$-stable*. A high-level description of our proposed framework is presented in Figure 5.1, in which $\alpha$-stable microaggregation is applied to generate $\overline{X}$ and $\overline{Y}$. Then $\varepsilon$-differentially private datasets $X_\varepsilon$ and $Y_\varepsilon$ are generated by applying perturbation over $\overline{X}$ and $\overline{Y}$, respectively.

Since there are $n/k$ pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$, the value of $\alpha$ ranges between $1 \leq \alpha \leq n/k$. Indeed we can see that insensitive microaggregation is a special case of $\alpha$-stable microaggregation with $\alpha = n/k$. This leads to the lemma below.

**Lemma 1.** *A $\alpha$-stable microaggregation algorithm $\mathcal{M}$ with $\alpha = n/k$ satisfy insensitive condition.*

As $\alpha$-stable microaggregation affects at most $\alpha$ pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$, the sensitivity $\Delta(f \circ \mathcal{M})$ of $\alpha$-stable microaggregation is $\alpha \times \Delta(f)/k$ as compared to $n/k \times \Delta(f)/k$ for insensitive microaggregation. However, to achieve $\Delta(f \circ \mathcal{M}) \leq \Delta(f)$ as desired, $(\alpha \times \Delta(f)/k) \leq \Delta(f)$ must hold. Therefore, one needs $k \geq \alpha$ for $\alpha$-stable microaggregation rather than $k \geq \sqrt{n}$ for insensitive microaggregation in order to reduce added noise in comparison with directly applying perturbation over $X$. Since, for $\alpha$-stable microaggregation, the corresponding sensitivity $\Delta(f \circ \mathcal{M}) = (\alpha \times \Delta(f)/k)$ directly depends on the value of $\alpha$, the addition of noise can always be controlled by $\alpha$ while generating $\varepsilon$-differentially private datasets to enhance utility. Moreover, $\alpha$ indicates the trade-off between information loss due to clustering during microaggregation and information loss due to noise addition in generating $\varepsilon$-differentially private datasets. As $\varepsilon$ indicates the privacy level in the process of achieving $\varepsilon$-differential privacy, the trade-off by $\alpha$ should be considered under a given $\varepsilon$.

On the other hand, to satisfy insensitive property a microaggregation algorithm requires the datasets $X$ and $Y$ to be equipped with a total order relation $\leq_X$. The distance function used by insensitive microaggregation algorithms must be consistent with a total order relation $\leq$ on $X$ [91]. The following lemma states the total order require to achieve insensitive property.

**Lemma 2.** *A microaggregation algorithm MDAV with distance $d : X \times X \rightarrow \mathbb{R}$ satisfies insensitive condition if and only if $d$ is consistent with an order relation $\leq_X$ such that for any records $x, y, z \in X$ if $d(x, y) \leq d(x, z)$ then it holds $x \leq_X y \leq_X z$.*

Due to imposed total order, insensitive microaggregation algorithm generates less homogeneous clusters and incurring more error during microaggregation as compared to standard (non-insensitive) microaggregation algorithm [89], such as MDAV [21]. However, in exchange for the lost of homogeneity, insensitive microaggregation generate sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ over $X \sim Y$ that are more stable than standard (non-insensitive) microaggregation when one record of the dataset changes due to the total order. Nevertheless, if a microaggregation algorithm does not comply with a total order then clusters in $\mathcal{C}_X$ may differ by more than one record from its corresponding cluster in $\mathcal{C}_Y$ and hence the algorithm would not satisfy insensitive property. Thus, total order is the key factor to satisfy insensitive property which results in degrading utility of differentially private datasets by providing less within cluster homogeneity over $X \sim Y$. This kind of predefined ordering over two neighboring dataset is a strong constraint, specifically when dataset is large and contains multiple attributes.

To alleviate this limitation, we require our proposed $\alpha$-stable microaggregation algorithms to be *consistent* with a partial order relation $\preceq_p$ on records which is stipulated by themselves when being applied to microaggregated neighboring datasets. We first describe our algorithms in Section 5.4 and then explain the partial order property posed by our algorithms can helps to achieve $\alpha$-stable property in Section 5.5.

## 5.4 $\alpha$-Stable Microaggregation Algorithms

Given two neighboring datasets $X \sim Y$ a user-defined parameter $\alpha$, our algorithms aim to generate clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ over $X$ and $Y$, respectively, such that the following two conditions are satisfied:

(i) There is a bijection between $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that at most $\alpha$ pairs of corresponding clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ differ in a single record, i.e., $\mathcal{C}_X$ and $\mathcal{C}_Y$ are $\alpha$-stable.

(ii) The objective function below is minimized:

$$\begin{aligned} &\textbf{minimize} \quad \sum_{c_j \in \mathcal{C}} \sum_{r_j \in c_j} d(r_j, r_{c_j}) && (5.1) \\ &\textbf{subject to} \quad \text{preserving a partial order relation } \preceq_p \text{ on records} \end{aligned}$$

Here, $d(r_i, r_j)$ is a distance function (e.g., Euclidean distance) which computes the distance between two records $r_i$ and $r_j$. By minimizing the distances between each record $r_j \in c_j$ and the representative record (centroid) $r_{c_j}$ of $c_j$ for each cluster $c_j \in \mathcal{C}$ as defined in Equation 5.1, the aim is to enhance within cluster homogeneity.

## 5.4.1 Sequential Search

To achieve $\alpha$-stable property while minimizing Equation 5.1, we propose an algorithm which takes a pair of neighboring datasets $X \sim Y$ and generates $\mathcal{C}_X$ with a standard microaggregation algorithm (e.g. MDAV [21]). Then based on $\mathcal{C}_X$ we generate $\mathcal{C}_Y$ such that clusters in $\mathcal{C}_X$ and $\mathcal{C}_Y$ are $\alpha$-stable yet homogeneous w.r.t. Equation 5.1. Suppose that a record $r$ in $X$ is modified to a record $r'$ in $Y$. We call the record $r'$ a *source record*. In order to generate $\mathcal{C}_Y$, the naive way is to search for a record $r^*$, by checking each record sequentially such that swapping a record $r'$ and a record $r^*$ leads to enhanced within cluster homogeneity. We call such a record $r^*$ as a *target record*. The key idea of this algorithm is to recursively carry out a record level search which finds a target record $r^*$ to swap with a source record $r'$ while preserving a partial ordering $\preceq_p$ in order to achieve $\alpha$-stable property and enhance within cluster homogeneity w.r.t. Equation 5.1.

For clarity, we use *source cluster* to refer to the cluster $c'$ that contains a source record $r'$, and similarly, *target cluster* to refer to the cluster $c_i$ that contains a target record $r_i$. We also use $z^+ = d(r_{c'}, r')$ to denote the distance between the representative record of a source cluster $r_{c'}$ and a source record $r'$, $z^- = d(r', r_{c_i})$ the distance between a source record $r'$ and the representative record of a target cluster $r_{c_i}$, $y^- = d(r_{c_i}, r_i)$ the distance between the representative record of a target cluster $r_{c_i}$ and a target record $r_i$, and $y^+ = d(r_i, r_{c'})$ the distance between a target record $r_i$ and the representative record of a source cluster $r_{c'}$. For a set of clusters $\mathcal{C}_Y$, we thus want to find a target cluster $c_i \in \mathcal{C}_Y$ and a target record $r^* \in c_i$, such that the following objective function is maximized:

$$r^* := \mathop{\textbf{argmax}}_{\substack{r_i \in c_i \\ c_i \in \mathcal{C}_Y}} \Big( (z^+ + y^-) - (z^- + y^+) \Big) \qquad (5.2)$$

$$\textbf{subject to} \quad \text{preserving a partial order relation } \preceq_p$$

**Algorithm 2:** *Sequential α-Stable Microaggregation*

```
Input: $X$: original dataset
       $(X, \preceq_p)$: a partial ordering
       $\mathcal{M} = (\mathcal{M}_p, \mathcal{M}_a)$ : a microaggregation algorithm
       $\alpha$: user defined parameter
Output: $\mathcal{C}_X, \mathcal{C}_Y$
1  $\mathcal{C}_X \leftarrow \{c_1, ..., c_n\}$ generated by $\mathcal{M}_p$ over $X$
2  $\mathcal{C}_Y \leftarrow Replace(\mathcal{C}_X, r, r')$ ;          // $X \sim Y$ where $r := X - Y$ and $r' := Y - X$
3  $c' \leftarrow FindSourceCluster(\mathcal{C}_Y, r')$
4  $\mathcal{C}_{search} := \mathcal{C}_Y \setminus c'$
5  $P_{active} := \{(c', r')\}$
6  $i := 1$
7  while $i < \alpha$ do
8      $W := \phi$
9      foreach $(c_i, r_i) \in P_{active}$ do
10         $(c_i^*, r_i^*, v_i) \leftarrow FindTargetRecord((c_i, r_i), \mathcal{C}_{search})$
11         $W := W \cup \{(c_i^*, r_i^*, c_i, r_i, v_i\}$
12     end
13     $c_i^*, r_i^*, c_i, r_i \leftarrow argmax(v_i)$
14     if $(c_i^*, r_i^*, c_i, r_i)$ exists then
15         if Order Preserved then
16             $\mathcal{C}_Y \leftarrow SwapRecords(c_i^*, r_i^*, c_i, r_i, \mathcal{C}_Y)$
17             $P_{active} := P_{active} \setminus \{(c_i, r_i)\} \cup \{(c_i^*, r_i), (c_i, r_i^*)\}$
18             $\mathcal{C}_{search} := \mathcal{C}_{search} \setminus \{c_i^*\}$
19             $UpdateOder(\preceq_p, r_i, r_i^*)$
20         end
21         $i := i + 1$
22     else
23         break
24     end
25 end
26 Return $\mathcal{C}_X, \mathcal{C}_Y$
```

The intuition behind maximizing the distance $(z^+ + y^-) - (z^- + y^+)$ is to find a target record $r^*$ and swap with source record $r'$, in such a way that can reduce the distance between each swapped record in a cluster and its cluster representative record to enhance within cluster homogeneity w.r.t. Equation 5.1.

Algorithm 2 provides the high level description of this algorithm. We refer Algorithm 2 as *sequential α-stable*. Given a dataset $X$, a partial ordering $\preceq_p$ (initially empty) over the domain of $X$, a microaggregation algorithm $\mathcal{M}$ = ($\mathcal{M}_p$, $\mathcal{M}_a$), where $\mathcal{M}_p$ and $\mathcal{M}_a$ refer to partition and aggregation functions of a microaggregation algorithm $\mathcal{M}$, respectively, and a user defined parameter $\alpha$. We start with partitioning the dataset $X$ into $\mathcal{C}_X$ by $\mathcal{M}_p$ and replace the record $r \in \mathcal{C}_X$ with $r'$ to initialize $\mathcal{C}_Y$, such that $X \sim Y$ where $r := X - Y$ and $r' := Y - X$ (Lines 1-2). Then, we find the source cluster containing $r'$, i.e., $c'$, by using *FindSourceCluster* function which takes a record as an input and return its corresponding cluster, and hence initialize

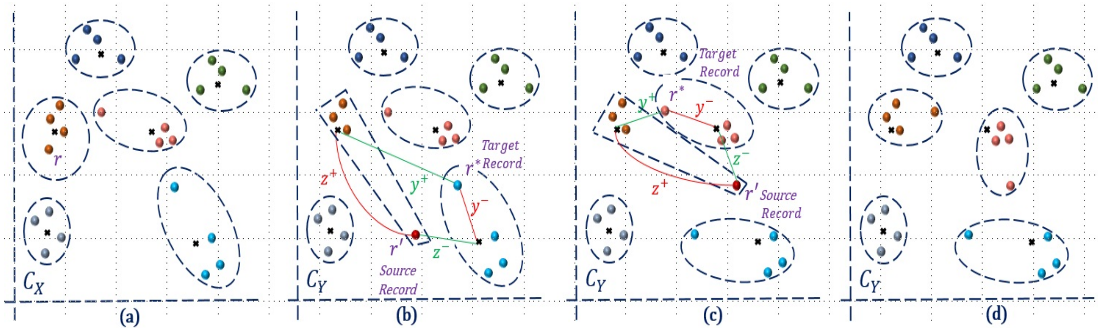


**Figure 5.2**: An illustration of *sequential α-stable microaggregation*.

the searching space $\mathcal{C}_{search}$ and the active pair list $P_{active}$, where each pair contains a record $r'$ and its corresponding cluster $c'$ (Lines 3-5). We also initialize an empty list $W$ (Line 8), then with help of *FindTargetRecord* function which takes a cluster record pairs in $P_{active}$, and searching space $\mathcal{C}_{search}$ as an input to compute the distance $v_i = (z^+ + y^-) - (z^- + y^+)$ by sequentially enumerating all the records in the searching space $\mathcal{C}_{search}$ and return a records $r_i^*$, its corresponding target cluster $c_i^*$, and the distance $v_i$ for each pair in $P_{active}$ (Lines 9-10). These results are stored in $W$ (Line 11). After building the list $W$ for all pairs, we select one record with the largest distance $v_i$ (must be a positive value), from $W$ (Lines 13-14). Then we check whether order is preserved or not (Line 5), if yes then we begin to update set of clusters $\mathcal{C}_Y$, the searching space $\mathcal{C}_{search}$, the active pair list $P_{active}$, and update ordering by extending the partial order with swapped records and begin next iteration, otherwise we skip swapping and begin next iteration (Lines 16-2). In the first step, we swap records $r_i^*$ and $r_i$ and update the microaggregated set of clusters $\mathcal{C}_Y$ (Line 16). Then we consider these two records and their corresponding clusters as new pairs and remove the previous pair $(c_i, r_i)$ from $P_{active}$ (Line 16). Finally, we update the searching space by excluding cluster $c_i^*$ (Lines 17), update partial ordering by extending it with $r_i$ and $r_i^*$ (i.e., $r_i \preceq_p r_i^*$) (Line 19) and begin the next iteration (Lines 20-21). The algorithm terminates in two conditions: (1) the sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ are $\alpha$-stable (Line 7); or (2) the distance of two records is negative, i.e. a target record $r_i^*$ does not exist (Lines 22-23). The algorithm returns $\mathcal{C}_X$ and $\mathcal{C}_Y$ as output (Line 26).

Here, we give an example to illustrate the sequential $\alpha$-stable microaggregation algorithm.

**Example 2.** *Consider Figure 5.2, for a pair of neighboring datasets $X \sim Y$ with $|X| = |Y| = n$. Let $\mathcal{M}$, with $\alpha = 3$ and $k = 4$, be a sequential $\alpha$-stable microaggregation algorithm which partitions $X$ into $\mathcal{C}_X$, as shown in Figure 5.2(a). Then to generate $\mathcal{C}_Y$, we initialize $\mathcal{C}_Y$ by replacing a record $r$ in $\mathcal{C}_X$ to a record $r'$, i.e., source record. As shown in Figure 5.2(b), suppose we have found a target record $r^*$ and ordered is preserved because initially its empty. Thus we swap the source record $r'$ and the target record $r^*$ and update partial ordering by extending it with $r'$ and $r^*$ (i.e., $r' \preceq_p r^*$). Then, we update the target record $r^*$ as the new source record $r'$ by updating searching space. As shown in Figure 5.2(c), we recursively find the next target record $r^*$ among the new searching space and swap with the source record $r'$*

*if ordered is being preserved, and keep on extending existing partial. For instance we have $r'$ initially and we swap it with $r^*$ and extend $\preceq_p$ which is $r' \preceq_p r^*$, then for next iteration we swap $r^*$ with $r_i^*$ because swap is consistent with $r' \preceq_p r^*$, thus we extend extend $\preceq_p$ which is $r' \preceq_p r^* \preceq_p r_i^*$. Figure 5.2(a), and (d) depicts the final sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ generated over $X$ and $Y$, respectively, by executing sequential α-stable microaggregation algorithm.*

**Proposition 1.** *Algorithm 2 is α-stable.*

*Proof.* Let $X$ and $Y$ be neighbouring data sets that are differ by only one record, that is, $r$ is in $X$ but not $Y$ and $r'$ is in $Y$ but not $X$. The algorithm starts by running a microaggregation sub-routine $\mathcal{M}_p$ on $X$ to get the clusters $\mathcal{C}_X$, and replacing $r$ with $r'$ to get a different set of clusters $\mathcal{C}_Y$ on $Y$. It entails that there is a bijection between the two sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$. Starting from the source record $r'$ in $Y$, the algorithm searches for a target record $r^*$ from the set $C_{search}$. Once a target record is found, we check whether swapping records preserve partial order if yes we extend the partial order. Each time when a target record is found, the cluster that contains the target record is removed from $C_{search}$. This guarantees each cluster in $\mathcal{C}_Y$ is differ from the corresponding cluster in $\mathcal{C}_X$ by only one record. Since the *While* loop iterates at most $\alpha - 1$ times, the final output of the algorithm cannot have more than $\alpha$ pairs clusters that that differ by one record. Hence, the algorithm satisfies $\alpha$-stable. □

**Complexity Analysis.** In order to find a target record the *sequential α-stable microaggregation algorithm* needs to perform sequential search over all records in $\mathcal{C}_{search}$. Considering a dataset of size $n$ and at least $k$ records in each cluster. Firstly, sequential $\alpha$-stable microaggregation algorithm needs to find the target record across all the records excluding ones in the source cluster having $r'$, hence we need to check the distance of $(n-k)$ record pairs. After selecting and swapping the records, we have 2 modified clusters and for the next iteration we need to search the rest $(n-2k)$ records for finding a target record. Hence, we need to calculate the distances of $2(n-2k)$ record. This process iterates, and for the last iteration, we have in total $\alpha - 1$ modified clusters and the total number of distances we need to check is $(\alpha-1)(n-(\alpha-1)k)$. Thus the total cost of Algorithm 2 is as follows:

$$= (n-k) + 2(n-2k) + 3(n-3k) + \cdots + (\alpha-1)(n-(\alpha-1)k)$$

Hence the complexity of *sequential α-stable microaggregation algorithm* is $\mathcal{O}(\alpha^2 n)$.

Although *sequential α-stable microaggregation algorithm* can generate clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ that are $\alpha$-stable with enhancing the homogeneity of these cluster, in terms of time complexity, it is expensive due to an exhaustive search for all records in $\mathcal{C}_Y$ to find a target record. If the size of a dataset and the value of $\alpha$ are large then it may take prohibitively long time to generate more homogeneous clusters. Thus to reduce time complexity, we introduced *decisional α-stable microaggregation algorithm* which approximates *sequential α-stable microaggregation algorithm.*

The rationale behind decisional $\alpha$-stable microaggregation is to search a target record $r^*$ at cluster level instead of looking into each record exhaustively. This enhance the performance of decisional $\alpha$-stable microaggregation as compare to sequential $\alpha$-stable microaggregation while achieving $\alpha$-stable property and enhancing within cluster homogeneity w.r.t. Equation 5.1. Thus for larger datasets, decisional $\alpha$-stable microaggregation algorithm takes less time than sequential $\alpha$-stable microaggregation algorithm to generate homogeneous clusters.

### 5.4.2 Decisional Search

The key idea of *decisional $\alpha$-stable microaggregation algorithm* is to recursively carry out a cluster level search which finds a target cluster $c^*$, and then finds a target record $r^*$ within the target cluster $c^*$ to swap with a source record $r'$ in order to achieve $\alpha$-stable property and enhance within cluster homogeneity w.r.t. Equation 5.1.

**Cluster Level Search.** Let $r'$ be a *source record*, $c'$ be a *source cluster* that contains $r'$, and $c_i$ be a *target cluster*. Now let $z^+ = d(r_{c'}, r')$ be the distance between the representative record of source cluster $r_{c'}$ and the source record $r'$, $z^- = d(r', r_{c_i})$ be the distance between the source record $r'$ and the representative record of target $r_{c_i}$, and $y^+ = d(r_{c_i}, r_{c'})$ be the distance between the representative record of target cluster $r_{c_i}$ and the representative record of source cluster $r_{c'}$. Given a set of clusters $\mathcal{C}_Y$, we want to find a target cluster $c_i \in \mathcal{C}_Y$, such that the following objective function is maximized:

$$\begin{aligned} c^* := \underset{c_i \in \mathcal{C}_Y}{\textbf{argmax}} & \ \Big(z^+ - (z^- + y^+)\Big) \\ \textbf{subject to} & \quad \text{preserving a partial order relation } \preceq_p \end{aligned} \tag{5.3}$$

The intuition behind maximizing the distance $(z^+ - (z^- + y^+)$ is to find a target cluster $c^*$ in such a way that can reduce the distance between a record in a cluster and its cluster representative record to enhance within cluster homogeneity w.r.t. Equation 5.1.

**Record Level Search.** After finding a target cluster $c^*$, the next step is to find a target record $r^* \in c^*$. In order to find a target record $r^* \in c^*$, let $z^+ = d(r_{c'}, r')$ be the distance between the representative record of a source cluster $r_{c'}$ and the source record $r'$, $z^- = d(r', r_{c^*})$ be the distance between the source record $r'$ and the representative record of a target cluster $r_{c^*}$, $y^- = d(r_{c^*}, r_i)$ be the distance between the representative record of a target cluster $r_{c^*}$ and a record $r_i \in c^*$, and $y^+ = d(r_i, r_{c'})$ be the distance between a record $r_i \in c^*$ and representative record of a source cluster $r_{c'}$. Given a target cluster $c^*$, we want to find a target record $r^* \in c^*$, such that the following objective function is maximized:

$$\begin{aligned} r^* := \underset{r_i \in c^*}{\textbf{argmax}} & \ \Big((z^+ + y^-) - (z^- + y^+)\Big) \\ \textbf{subject to} & \quad \text{preserving a partial order relation } \preceq_p \end{aligned} \tag{5.4}$$

**Algorithm 3:** *Decisional α-Stable Microaggregation*

```
Input: $X$: original dataset
       $(X, \preceq_p)$: a partial ordering
       $\mathcal{M} = (\mathcal{M}_p, \mathcal{M}_a)$ : a microaggregation algorithm
       $\alpha$: user defined parameter
Output: $\mathcal{C}_X, \mathcal{C}_Y$
1  $\mathcal{C}_X \leftarrow \{c_1, ..., c_n\}$ generated by $\mathcal{M}_p$ over $X$
2  $\mathcal{C}_Y \leftarrow Replace(\mathcal{C}_X, r, r')$ ;          // $X \sim Y$ where $r := X - Y$ and $r' := Y - X$
3  $c' \leftarrow FindSourceCluster(\mathcal{C}_Y, r')$
4  $\mathcal{C}_{search} := \mathcal{C}_Y \setminus c'$
5  $P_{active} := \{(c', r')\}$
6  $i := 1$
7  while $i < \alpha$ do
8      $W := \phi$
9      foreach $(c_i, r_i) \in P_{active}$ do
10         $(c_i^*, v_i) \leftarrow FindTargetCluster((c_i, r_i), \mathcal{C}_{search})$
11         $W := W \cup \{(c_i^*, c_i, r_i, v_i)\}$
12     end
13     $c_i^*, c_i, r_i \leftarrow argmax(v_i)$
14     $r_i^* \leftarrow FindTargetRecord((c_i, r_i), c_i^*)$
15     if $r_i^*$ exists then
16         if Order Preserved then
17             $\mathcal{C}_Y \leftarrow SwapRecords(c_i^*, r_i^*, c_i, r_i, \mathcal{C}_Y)$
18             $P_{active} := P_{active} \setminus \{(c_i, r_i)\} \cup \{(c_i^*, r_i), (c_i, r_i^*)\}$
19             $\mathcal{C}_{search} := \mathcal{C}_{search} \setminus \{c_i^*\}$
20             $UpdateOder(\preceq_p, r_i, r_i^*)$
21         end
22         $i := i + 1$
23     else
24         break
25     end
26 end
27 Return $\mathcal{C}_X, \mathcal{C}_Y$
```

If there exists a record $r_i \in c^*$ which can maximizes the distance $(z^+ + y^-) - (z^- + y^+)$ and can also preserve the existing order, then such a record is selected as the target record, i.e, $r^*$. The intuition behind maximizing the distance $(z^+ + y^-) - (z^- + y^+)$ is is to find a target record $r^*$ in a target cluster $c^*$ and swap with source record $r'$, in such a way that can reduce the distance between each swapped record in a cluster and its cluster representative record to enhance within cluster homogeneity w.r.t. Equation 5.1. Algorithm 3 provides the high level description of this algorithm.

Here we briefly illustrate *decisional α-stable microaggregation algorithm* with a focus on the improvements compared with *sequential α-stable microaggregation algorithm*. We start with initializing $\mathcal{C}_X$, $\mathcal{C}_Y$ (Lines 1-2). Then, we find the source cluster to initialize the searching space $\mathcal{C}_{search}$ and the active pair list $P_{active}$ (Lines 3-5). Then we start to search for the target cluster for each pair in $P_{active}$ via the representative record $r_{c_i}$ of

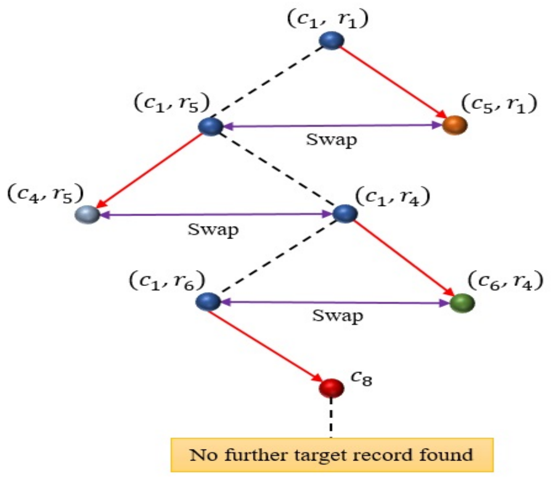


**Figure 5.3**: An illustration of *decisional $\alpha$-stable microaggregation*.

cluster $c_i$ instead of sequentially enumerating all the records in the searching space $\mathcal{C}_{search}$ using the function *FindTargetCluster*(Lines 9-10). After selecting the cluster $c_i^*$ with the largest distance, we search for the target record $r_i^*$ and select one record with the largest distance $v_i$ (must be a positive value) using the function *FindTargetRecord* (Lines 11-14). If the target record exists (Line 15) then we check whether the order is being preserved or not (Line 16), if yes we continue to update clusters by swapping records $r_i^*$ and $r'$ (Line 17). Then, we update the active pair list $P_{active}$, the searching space $\mathcal{C}_{search}$, and update ordering by extending the partial order with swapped records and begin next iteration, otherwise we skip swapping and begin next iteration (Lines 17-22). The algorithm terminates in two conditions: (1) the sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ are $\alpha$-stable (Line 7); or (2) the distance of two records is negative, i.e. a target record $r_i^*$ does not exist (Lines 23-24). The algorithm returns $\mathcal{C}_X$ and $\mathcal{C}_Y$ as output (Line 27).

Here, we give an example to illustrate the decisional $\alpha$-stable microaggregation algorithm.

**Example 3.** *Consider Figure 5.3, in which $\mathcal{C}_Y = \{c_1, c_2, \ldots, c_8\}$ be a set of clusters. Suppose $c_1$ be a source cluster that contains a source record $r_1$. Then to perform a cluster level search, we initialize $\mathcal{C}_{search} = \mathcal{C}_Y \setminus c_1$ and $P_{active} = \{(c_1, r_1)\}$. Suppose, for every pair in $P_{active}$, we have found $c_5$ as a target cluster in the first iteration using Equation 5.3. Once the cluster is decided, then we need to perform a record level search to find the target record as mentioned in Equation 5.4. Suppose we have $r_5 \in c_5$ as a target record. If the order is preserved, then we need to swap found records in $c_1$ and $c_5$ as depicted in Figure 5.3 and update $P_{active} = \{(c_1, r_5), (c_5, r_1)\}$. For the next iteration we remove $c_5$ from $\mathcal{C}_{search}$ and perform cluster level search in $\mathcal{C}_{search}$ to find the target cluster for each pair in $P_{active}$. Suppose in the following iterations we have found $c_4$, $c_6$ and $c_8$ as the target clusters respectively and suppose the order remains preserved. Algorithm 3 recursively performs cluster level and record level searches until $\alpha$ reaches or there exists no target cluster with target record.*

**Proposition 2.** *Algorithm 3 is $\alpha$-stable.*

*Proof.* The algorithm performs cluster level search in $\mathcal{C}_{search}$ to find the target cluster, and then perform a record level search to find the target record within target cluster. Once a target record is found, we check whether swapping records preserve partial order if yes we extend the partial order. Each time when a target record is found, same as Algorithm 2, Algorithm 3 removes the cluster that contains the target record from $C_{search}$. This guarantees each cluster in $\mathcal{C}_Y$ is differ from the corresponding cluster in $\mathcal{C}_X$ by only one record. Since the *While* loop iterates at most $\alpha - 1$ times, the final output of the algorithm cannot have more than $\alpha$ pairs clusters that that differ by one record. Hence, the algorithm satisfies $\alpha$-stable. □

**Complexity Analysis.** In order to find a target cluster, the *decisional $\alpha$-stable microaggregation algorithm* needs to perform cluster level search over all clusters in $\mathcal{C}_{search}$ and then record level search over all records in the target cluster. Considering a dataset of size $n$ and at least $k$ records in each cluster. Firstly, decisional $\alpha$-stable microaggregation algorithm needs to find the target cluster across all the clusters excluding the modified cluster, then within this cluster we need to find the target record, hence we need to check the distance of $(\frac{n}{k} - 1)$ clusters with at least $k$ records in total, i.e. $(\frac{n}{k} - 1) + k$. After selecting and swapping the records, we have 2 modified clusters and for the next iteration we need to search the rest $(\frac{n}{k} - 2)$ clusters for the target record. Hence, we need to calculate the distances of $2(\frac{n}{k} - 2)$ clusters, as well as $k$ records within the target cluster. This process iterates, and for the last iteration, we have in total $\alpha - 1$ modified clusters and the total number of distances we need to check is $(\alpha - 1)(\frac{n}{k} - (\alpha - 1)) + k$. Thus the total cost of Algorithm 3 is as follows:

$$= (\frac{n}{k} - 1) + k + 2(\frac{n}{k} - 2) + k + \cdots + (\alpha - 1)(\frac{n}{k} - (\alpha - 1)) + k$$

Hence, the complexity of *decisional $\alpha$-stable microaggregation algorithm* is $\mathcal{O}(\alpha k + \alpha^2 \frac{n}{k})$.

The complexity for *sequential $\alpha$-stable microaggregation algorithm* is $\mathcal{O}(\alpha^2 n)$ as it searches for all records in $\mathcal{C}_Y$. Instead the complexity for *decisional $\alpha$-stable microaggregation algorithm* is $\mathcal{O}(\alpha k + \alpha^2 \frac{n}{k})$ as it searches for all clusters in $\mathcal{C}_Y$ and then search for all $k$ records of the respective cluster. Thus, when $\alpha > 1$ and $k > 1$, we can say that the complexity of decisional $\alpha$-stable microaggregation algorithm is always lower as compared with sequential $\alpha$-stable microaggregation algorithm. In the best case, the complexity of decisional $\alpha$-stable microaggregation algorithm is reduced by at most $k$. For larger datasets, decisional $\alpha$-stable microaggregation algorithm takes less time than sequential $\alpha$-stable microaggregation algorithm to generate more homogeneous clusters.

## 5.5 Order property

Let $\preceq_p$ be a partial order relation among records over the domain of $X$. The proposed algorithms need to be *consistent* with the partial order relation $\preceq_p$ to satisfy $\alpha$-stable property.

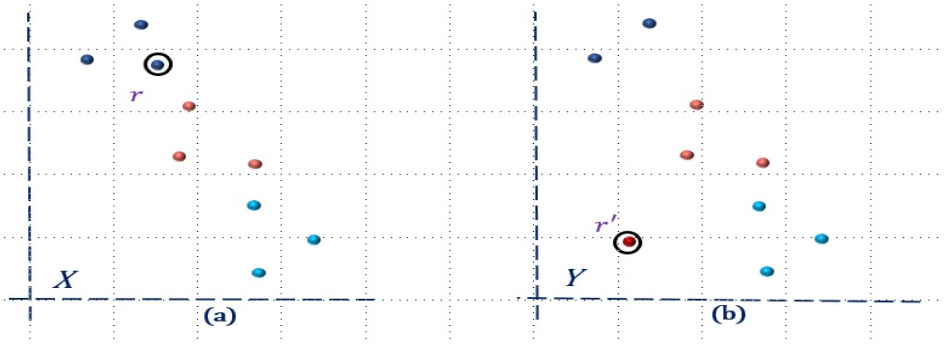


**Figure 5.4**: Neighboring datasets $X$ and $Y$.

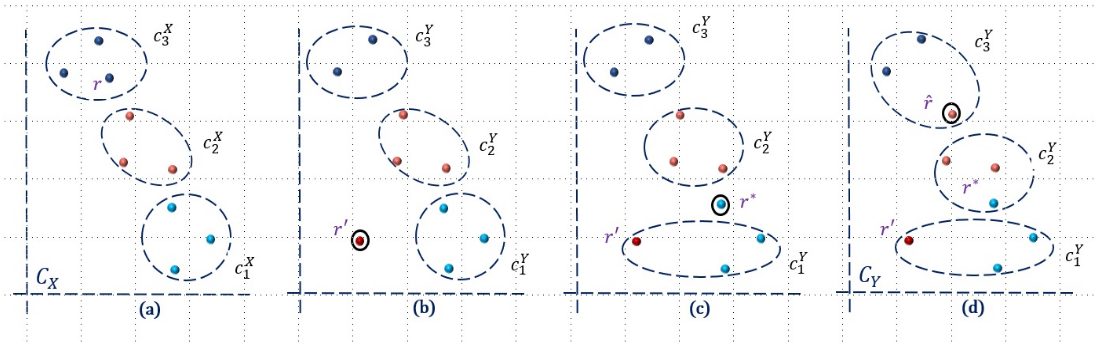


**Figure 5.5:** An illustration of a partial order for $X \sim Y$: (a) show sets of clusters $\mathcal{C}_X$ over $X$, (b) and (c) describes swapping operation of *α-stable microaggregation* algorithms which define partial ordering to generate $\mathcal{C}_Y$, and (d) show sets of clusters $\mathcal{C}_Y$ over $Y$ with a partial ordering $\preceq_p$.

**Definition 15.** ***(Consistent)*** *Given a set of clusters $\mathcal{C} = \{c_1, \ldots, c_m\}$, any two clusters are said to be consistent with a partial order relation $\preceq_p$ (i.e., $c_1 \preceq_p c_2$), if $\forall_{r_i \in c_1}$ and $\forall_{r_j \in c_2}$ it holds that $r_i \preceq_p r_j$.*

**Example 4.** *Let $X \sim Y$ be two neighboring datasets as shown in Figure 5.4, and $\mathcal{C}_X$ be a set of clusters generated by a standard microaggregation algorithm $\mathcal{M}$ (e.g., MDAV) as depicted in Figure 5.5(a). Suppose that a record $r$ in $X$ is modified to $r' =$ in $Y$ as shown in Figure 5.5(b). Consider Figure 5.5(c) and (d), and let α-stable $\mathcal{M}$ be a microaggregation algorithm that achieves α-stable property and enhances within cluster homogeneity w.r.t Equation 5.1 via recursively shifting records in $\mathcal{C}_Y$. Suppose in the first recursive call, as shown in Figure 5.4(c), we have found a target record $r^*$ and swap the source record $r'$ and the target record $r^*$. Then, in the next recursive call we have found next target record, lets call it $\hat{r}$, and swap $\hat{r}$ and $r^*$. Thus, we have a partial ordering $r' \preceq_p r^* \preceq_p \hat{r}$ at record level, where $r' \in c_1^Y$, $r^* \in c_2^Y$, and $\hat{r} \in c_3^Y$. According to Definition 15 we can say that there exist a partial order $c_1^Y \preceq_p c_2^Y \preceq_p c_3^Y$, where $\forall_{r_i \in c_1^Y}, \forall_{r_j \in c_2^Y}, \forall_{r_k \in c_3^Y}$, we have $r_i \preceq_p r_j \preceq_p r_k$. For instance, if $c_1^Y = \{r', r_1, r_2\}$ and $c_2^Y = \{r^*, r_3, r_4\}$, then we have a partial order $r' \preceq_p r^*$, $r' \preceq_p r_3$, $r' \preceq_p r_4$, $r_1 \preceq_p r^*$, $r_1 \preceq_p r_3, r_1 \preceq_p r_4$, $r_2 \preceq_p r^*$, $r_2 \preceq_p r_3, r_2 \preceq_p r_4$ between $c_1^Y \preceq_p c_2^Y$. By*

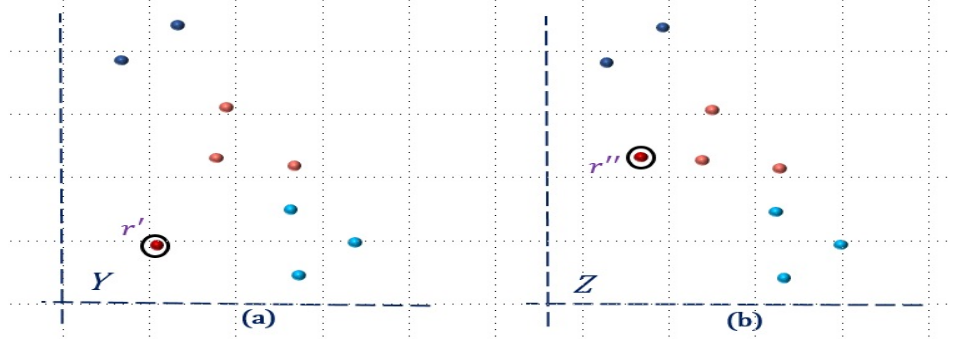


**Figure 5.6**: Neighboring datasets $Y$, and $Z$.

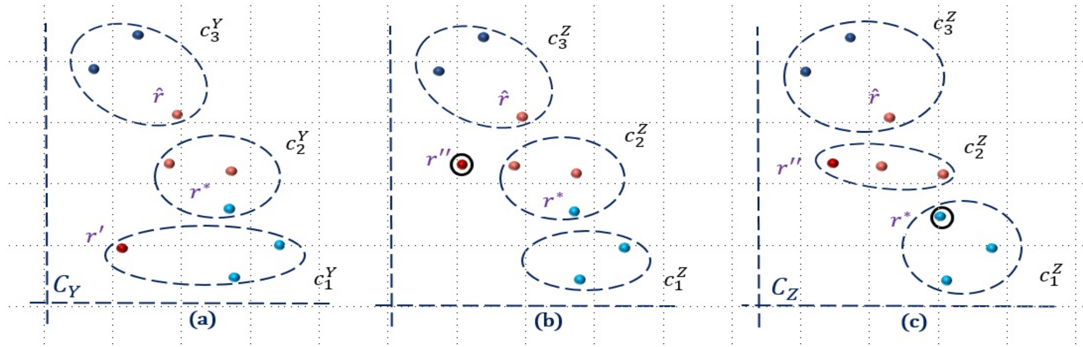


**Figure 5.7**: An illustration of swapping by preserving a partial ordering.

*extending such a partial ordering, it can easily lead to a total ordering. Figure 5.4(a), and (d) depicts the final sets of clusters $\mathcal{C}_X$ and $\mathcal{C}_Y$ generated by $\alpha$-stable $\mathcal{M}$.*

Now suppose $X \sim Y \sim Z$ be three neighboring datasets such that, $X \sim Y$, $Y \sim Z$, and $X \sim Z$ as shown in Figures 5.4 and 5.6. Here we show that, for each neighboring dataset pair, our proposed algorithms extend a partial order in a way that it must respect/preserve the existing partial order to swap records between clusters and should also add new partial ordering posed during the execution of the algorithms.

**Example 5.** *Consider Figure 5.7. Given a partial order $r' \preceq_p r^* \preceq_p \hat{r}$ as shown in Figure 5.7(a). Let $r'$ in $Y$ is modified to $r''$ in $Z$ as shown in Figure 5.7(b). The $\alpha$-stable microaggregation algorithm $\mathcal{M}$ needs to recursively shift records in $\mathcal{C}_Z$ to enhance within cluster homogeneity w.r.t Equation 5.1. However, $\mathcal{M}$ requires to* respect *the existing partial order. Suppose for $r''$ we have found a record $r^*$, as shown in Figure 5.7(c). We only swap these records if and only if the swapping is* consistent *with the existing partial order, otherwise we skip swapping and continue our search. In this example we swap the records $r''$ and $r^*$ as given partial order (i.e., $r^* \preceq_p \hat{r}$) is preserved. Thus, we have an updated partial order $r^* \preceq_p r'' \preceq_p \hat{r}$, where $r^* \in c_1^Z$, $r'' \in c_2^Z$, and $\hat{r} \in c_3^Z$. According to Definition 15 we can say that there exist a partial order $c_1^Z \preceq_p c_2^Z \preceq_p c_3^Z$. Figure 5.7(a), and (c) depicts the final sets of clusters $\mathcal{C}_Y$, and $\mathcal{C}_Z$ generated by $\alpha$-stable $\mathcal{M}$ by respecting the partial order $\preceq_p$.*

**Proposition 3.** *Algorithms 2, and 3 satisfy $\alpha$-stable condition if and only if there exist a partial order relation $\preceq_p$ among the records in $X \sim Y$, and Algorithms 2, and 3 swap records in clusters in a way that is consistent with $\preceq_p$; further $\preceq_p$ is extended by swapped records in clusters.*

*Proof.* Given a dataset $X$, we want to check that Algorithms 2, and 3 requires partial order relation $\preceq_p$ among the records in $X \sim Y$ to satisfy $\alpha$-stable condition, i.e., modifying a single record of $X$ leads to the set of clusters such that at most $\alpha$ pairs of corresponding clusters $< (c, c'), (c_1, c'_1), (c_2, c'_2), \ldots, (c_{(\alpha-1)}, c'_{(\alpha-1)}) >$ differ in at most one record.

Suppose we modify a record $r$ in $X$ by setting it to $r'$, and let $Y$ be the modified dataset (i.e., $X \sim Y$). If $\alpha = 1$, it is a trivial which shows that there exists a bijection between $\mathcal{C}_X$ and $\mathcal{C}_Y$ such that one pair of corresponding clusters $(c, c')$ differ in a single record, as $r \in c$ in $\mathcal{C}_X$ and $r' \in c'$ in $\mathcal{C}_Y$. Thus, $c$ and $c'$ differ in single record, therefore $X \setminus c$ and $Y \setminus c'$ are equal.

Let $c, c_1, \ldots, c_{(\alpha-1)}$ and $c', c'_1, \ldots, c_{(\alpha-1)}$ be, respectively, the clusters of $X$ and $Y$, ordered according to $\preceq_p$. If $2 \leq \alpha \leq n/k$, we want to show that for any $1 \leq i \leq (\alpha - 1)$, the clusters $c_i$ and $c'_i$ differ in at most one record. Cluster $c'_1$ contains same records as $c_1$ except for $r'_1$ that has been removed from $c'_1$ and for $r'$, such that $r' \preceq_p r'_1$, that has been added to $c'_1$, i.e., $c'_1 = (c_1 \cup \{r'\} \setminus \{r'_1\})$. Similarly, clusters $c'_2, \ldots, c'_{(\alpha-1)}$ contain same records as respective cluster $c_2, \ldots, c_{(\alpha-1)}$, except for $r'_i$ that has been removed from $c'_i$ and for $r'_{i-1}$, such that $r'_{i-1} \preceq_p r'_i$, that has been added to $c'_i$. Therefore, clusters $c_i$ and $c'_i$ differ in single record for all $i$, and $X \setminus \{c, c_1, \ldots c_{(\alpha-1)}\}$ and $Y \setminus \{c', c'_1, \ldots c'_{(\alpha-1)}\}$ are equal, which completes the proof. $\square$

In Proposition 3, we have shown that Algorithm 2, and 3 are $\alpha$-stable, when a partial order relation $\preceq_p$ is preserved and extended over $X \sim Y$. By Proposition 3, it is straight forward to check that Algorithms 2, and 3 are insensitive, as there are $n/k$ pairs of corresponding clusters, therefore, the value of $\alpha$ ranges between $1 \leq \alpha \leq n/k$. Also extending a partial order will leads to a total order which is a must condition to satisfy insensitive property. Indeed we can see that insensitive microaggregation becomes a special case of $\alpha$-stable microaggregation with $\alpha = n/k$, and Algorithms 2, and 3 can be reduced to insensitive microaggregation algorithm when partial order leads to total order.

## 5.6 Experiments

In this section, I presented the experiments of evaluating the proposed microaggregation-based algorithms *sequential $\alpha$-stable microaggregation* (S-$\alpha$S-MDAV) and *decisional $\alpha$-stable microaggregation* (D-$\alpha$S-MDAV) against the state-of-the-art methods. The experiments aim to answer the following research questions:

- **RQ1:** Do S-$\alpha$S-MDAV and D-$\alpha$S-MDAV yield less information loss by providing better within cluster homogeneity in microaggregated datasets and reducing noise added to differentially private datasets as compared to state-of-the-art methods?

- **RQ2:** Do S-$\alpha$S-MDAV and D-$\alpha$S-MDAV yield more privacy by reducing probability of correct record linkages in microaggregated datasets and reducing disclosure risk to differentially private datasets as compared to state-of-the-art methods?

- **RQ3:** What kind of trade-off exists between utility and privacy while generating differentially private datasets for different values of $\alpha$, $k$ and $\varepsilon$?

### 5.6.1 Experimental Setup

**Datasets.** I used three well-established datasets. These datasets have been used in the European project "CASC" and since then, have become references to test and compare statistical disclosure control methods for protecting numerical microdata.

- **CENSUS**[1]**:** This dataset contains 1,080 records [91; 21; 89]. As in [91] we took 4 numerical attributes: FEDTAX (Federal income tax liability), FICA (Social security retirement payroll deduction), INTVAL (Amount of interest income) and POTHVAL (Total other persons income).

- **EIA**[1]**:** This dataset contains 4,092 records [18; 61; 46; 13]. As in [46] we took 4 numerical attributes: RESREVENUE (Revenue from sales to residential consumers), RESSALES (Sales to residential consumers), TOTREVENUE (Revenue from sales to all consumers), and TOTSALES (sales to all consumers).

- **Tarragona**[1]**:** This dataset contains 834 records [61; 18; 19]. We took 5 numerical attributes: Fixed assets, Current assets, Paid-up capital, Short-term debt, and Sales.

**Baseline Methods.** In order to evaluate the proposed framework, I considered the following baseline methods:

- MDAV, which is a standard microaggregation algorithm [21].

- I-MDAV, which is an insensitive microaggregation algorithm proposed in [91].

- $\varepsilon$-DP, which is a standard $\varepsilon$-differential privacy algorithm in which noise is added using the Laplace mechanism [29].

- S-$\alpha$S-MDAV and D-$\alpha$S-MDAV refer to the proposed sequential $\alpha$-stable microaggregation and decisional $\alpha$-stable microaggregation algorithms respectively. Both of these algorithms extend the standard microaggregation algorithm MDAV [21] in partitioning and aggregation.

[1]http://neon.vb.cbs.nl/casc/CASCtestsets.htm

**Evaluation Measures.** Since the proposed work aims at generating differentially private datasets via *k*-anonymous microaggregation, I have used the evaluation measures used by the *k*-anonymity community [91; 19; 21]. In such works, the quality of an anonymized dataset is calculated in terms of *information loss*, which directly affects the utility of a dataset, and *disclosure risk*, which measures the practical privacy of a dataset:

- *Information loss* measures the difference between an original and anonymized datasets. I used the measure $IL1s$ [74; 109] to compute the *information loss* between the original and differentially private datasets. Formally, for each record $r_i$,

$$IL1s = \frac{1}{|A| \cdot n} \sum_{i=1}^{n} \sum_{j=0}^{|A|} \frac{|x_{ij} - x'_{ij}|}{\sqrt{2}S_j} \tag{5.5}$$

  where $|A|$ is the number of attributes, $n$ is the number of records in the dataset, $x_{ij}$ is the value of attribute $a_j \in A$ for record $r_i$ in the original dataset, $x'_{ij}$ is the value of attribute $a_j \in A$ for record $r_i$ in the corresponding differentially private dataset, and $S_j$ is the standard deviation of attribute $a_j \in A$ in the original dataset. Thus, with higher information loss, the utility of the anonymized dataset is lower.

- *Disclosure risk* has been evaluated as the percentage of Record Linkages (RL), i.e., the percentage of records in the original dataset that can be correctly matched with an anonymized dataset.

$$RL = 100 \times \frac{\sum_{r_i \in X} Pr(r'_i)}{n} \tag{5.6}$$

  where $n$ is the number of records in the dataset $X$, $r_i$ is an original record, and the probability of record linkage for the anonymized record $Pr(r'_i)$ is calculated as

$$Pr(r'_i) = \begin{cases} 0 & \text{if } r_i \notin R \\ \frac{1}{|R|} & \text{if } r_i \in R \end{cases} \tag{5.7}$$

  where $R$ is the set of original records that are at minimum distance from $r'_i$. We have used Euclidean distance. With lower percentage of RL, the probability of identity disclosure is lower. Therefore, the privacy of the anonymized dataset is higher with lower percentage of RL.

**Parameter Settings and Others.** Following [91], I considered the sensitivity of an attribute to be the difference between the lower bound (i.e. 0) and upper bound (1.5 $\times$ the maximum value) of the attribute. For CENSUS, EIA and Tarragona datasets, the value of $k$ is set to between 2 and 100. For generating differentially private datasets, I used the following privacy parameters $\varepsilon = [0.01, 0.1, 1.0, 10.0]$, which cover the range of differential privacy levels widely used in the literature [28; 70; 89]. For each parameter setting of $\varepsilon$, I ran 3 times and took the average result. For generating differentially private datasets with S-$\alpha$S-MDAV and D-$\alpha$S-MDAV, I used the following parameters $\alpha = [2, 4, 6, 8, n/k]$, which cover the range of $\alpha$ for every value of $n$ and $k$ in CENSUS, EIA and Tarragona datasets.

**Table 5.1:** Comparison on the IL1s of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over CENSUS dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 191.04 | 190.33 | 190.36 | 190.22 | 189.76 | 190.22 | 189.4 | 190.22 | 189.51 | 190.22 | 398.08 |
| 20 | 611.08 | 610.165 | 611.08 | 610.37 | 611.08 | 610.37 | 611.08 | 610.37 | 611.08 | 610.37 | 757.86 |
| 40 | 770.08 | 771.062 | 770.16 | 771.06 | 770.16 | 771.06 | 770.16 | 771.06 | 770.16 | 771.06 | 893.37 |
| 60 | 864.82 | 868.051 | 864.92 | 868.47 | 865.6 | 868.47 | 866.09 | 868.47 | 868.94 | 868.47 | 1010.7 |
| 80 | 958.34 | 960.153 | 956.13 | 960.4 | 956.9 | 960.4 | 958.64 | 960.4 | 959.25 | 960.4 | 1130.6 |
| 100 | 1038.5 | 1040.51 | 1039.9 | 1040.5 | 1039.5 | 1040.5 | 1039.9 | 1040.5 | 1039.3 | 1040.5 | 1243.9 |

**Table 5.2:** Comparison on the %RL of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over CENSUS dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 34.7 | 34.5 | 34.72 | 34.4 | 34.7 | 34.4 | 34.9 | 34.4 | 35 | 34.4 | 14.3 |
| 20 | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.44** | **4.26** |
| 40 | 2.31 | 2.31 | 2.315 | 2.31 | 2.31 | 2.31 | 2.31 | 2.31 | 2.31 | 2.31 | 2.31 |
| 60 | 1.57 | 1.57 | 1.574 | 1.57 | 1.57 | 1.57 | 1.57 | 1.57 | 1.57 | 1.57 | 1.67 |
| 80 | 1.2 | 1.2 | 1.204 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.11 |
| 100 | 0.93 | 0.93 | 0.926 | 0.93 | 0.93 | 0.93 | 0.83 | 0.93 | 0.83 | 0.93 | 0.93 |

**Table 5.3:** Comparison on the IL1s of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over EIA dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 159.57 | 337.5 |
| 20 | 605.7 | 605.63 | 606.05 | 605.63 | 606.36 | 605.63 | 607.19 | 605.63 | 643.78 | 605.63 | 772.2 |
| 40 | 794.53 | 794.53 | 794.36 | 794.53 | 795.14 | 794.53 | 795.88 | 794.53 | 814.28 | 794.53 | 910.78 |
| 60 | 927.44 | 928.06 | 927.44 | 928.06 | 927.44 | 928.06 | 927.44 | 928.06 | 927.44 | 928.06 | 1050.3 |
| 80 | 992.69 | 992.55 | 992.74 | 992.55 | 992.76 | 992.55 | 993 | 992.55 | 1006 | 992.55 | 1149.5 |
| 100 | 1082.5 | 1083.8 | 1084.1 | 1083.8 | 1087.2 | 1083.8 | 1091.9 | 1083.8 | 1091.9 | 1083.8 | 1198 |

**Table 5.4:** Comparison on the %RL of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over EIA dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 25.5 | 12.3 |
| 20 | **3.54** | **3.52** | **3.54** | **3.52** | **3.57** | **3.52** | **3.59** | **3.52** | **3.54** | **3.52** | **2.76** |
| 40 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.81 | 1.78 | 1.81 | 1.71 |
| 60 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.44 | 1.25 |
| 80 | 1.17 | 1.17 | 1.17 | 1.17 | 1.17 | 1.17 | 1.17 | 1.17 | 1.12 | 1.17 | 0.95 |
| 100 | 0.93 | 0.93 | 0.93 | 0.93 | 0.93 | 0.93 | 0.9 | 0.93 | 0.9 | 0.93 | 0.9 |

**Table 5.5:** Comparison on the IL1s of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over Tarragona dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 202.46 | 199.44 | 200.41 | 199.44 | 200.41 | 199.44 | 200.41 | 199.44 | 200.41 | 199.44 | 288.88 |
| 20 | 550.08 | 560.99 | 550.01 | 561.39 | 548.66 | 561.39 | 549.29 | 561.39 | 558.53 | 561.39 | 560.15 |
| 40 | 646.24 | **657.87** | 643.53 | **657.87** | 643.53 | **657.87** | 643.53 | **657.87** | 643.53 | **657.87** | **646.96** |
| 60 | 720.19 | 730.58 | 720.77 | 730.58 | 727.6 | 730.58 | 728.17 | 730.58 | 727.72 | 730.58 | 697.34 |
| 80 | 761.89 | 761.89 | 761.89 | 761.86 | 762.12 | 761.86 | 762.44 | 761.86 | 762.4 | 761.86 | 740.63 |
| 100 | 792.64 | 792.64 | 797.79 | 794.56 | 797.87 | 794.56 | 798.43 | 794.56 | 798.43 | 794.56 | 783.59 |

**Table 5.6:** Comparison on the %RL of S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under different values of k and $\alpha$ over Tarragona dataset.

| **k** | $\alpha = 2$ | | $\alpha = 4$ | | $\alpha = 6$ | | $\alpha = 8$ | | $\alpha = n/k$ | | I-MDAV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | S-$\alpha$S-MDAV | D-$\alpha$S-MDAV | |
| 2 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 29.9 | 18.1 |
| 20 | **4.08** | **4.32** | **3.84** | **4.32** | **3.84** | **4.32** | **3.84** | **4.32** | **3.72** | **4.32** | **3** |
| 40 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 2.28 | 1.92 |
| 60 | 1.44 | 1.44 | 1.32 | 1.44 | 1.32 | 1.44 | 1.32 | 1.44 | 1.2 | 1.44 | 1.44 |
| 80 | 1.08 | 1.08 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 1.2 |
| 100 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.96 |

### 5.6.2 Experimental Results and Discussion

In this section, I presented the experimental results and discussed our observations.

**Utility.** I first conducted experiments to compare the information loss of microaggregated datasets that are generated by S-$\alpha$S-MDAV, D-$\alpha$S-MDAV and I-MDAV under varying $k$ between 2 to 100. The results are shown in Tables 5.1, 5.3, and 5.5. I observed that, for the datasets CENSUS and EIA, the information loss of microaggregated datasets for every value of $\alpha$ and $k$ is less with S-$\alpha$S-MDAV and D-$\alpha$S-MDAV as compared to I-MDAV. As I used MDAV in our algorithms S-$\alpha$S-MDAV and D-$\alpha$S-MDAV to generate clusters in $C_X$ as well as $C_Y$. This is because the clusters generated by MDAV are more homogeneous than the clusters generated by I-MDAV. However in Tarragona dataset, as highlighted, information loss of microaggregated datasets for larger values of $k$, (i.e., $k \geq 40$) is slightly more with S-$\alpha$S-MDAV and D-$\alpha$S-MDAV as compared to I-MDAV. This is because the homogeneity of clusters generated by I-MDAV is better when the dataset is small as compared with MDAV. Nonetheless, S-$\alpha$S-MDAV and D-$\alpha$S-MDAV decrease the sensitivity of $f \circ \mathcal{M}$ regardless of the size of a dataset and thus reduce the errors caused by microaggregation.

Then, to verify the overall utility of $\varepsilon$-differentially private datasets, I conducted experiments to compare the information loss between the original and $\varepsilon$-differentially private datasets generated using our algorithms S-$\alpha$S-MDAV, D-$\alpha$S-MDAV and the baseline methods I-MDAV and $\varepsilon$-DP. However, the results obtained by S-$\alpha$S-MDAV and D-$\alpha$S-MDAV are quite similar, thus for the purpose of clear visualization I only presented the results obtained by D-$\alpha$S-MDAV. Figures 5.8, 5.10 and 5.12 presented our experimental results. The information loss for $\varepsilon$-DP is displayed as horizontal lines, as $\varepsilon$-DP does not depend on $k$. Regarding the information loss shown in Figures 5.8, 5.10 and 5.12, it can be seen that, for every value of $\varepsilon$, I-MDAV is only able to achieve $\Delta(f \circ \mathcal{M}) \leq \Delta(f)$ if $k \geq \sqrt{n}$, i.e., ($k = \sqrt{1,080} \approx 33$ for CENSUS, $k = \sqrt{4,092} \approx 64$ for EIA, and $k = \sqrt{834} \approx 28$ for Tarragona). This is consistent with the previous discussion in Section 5.3. Nonetheless, this also means that for large datasets I-MDAV requires $k$ to be large in order to effectively reduce $\Delta(f \circ \mathcal{M})$.

In contrast, for S-$\alpha$S-MDAV and D-$\alpha$S-MDAV, as stated in Section 5.3, one needs $k \geq \alpha$ to reduce $\Delta(f \circ \mathcal{M})$ as compared to $\varepsilon$-DP. For values $\varepsilon = [0.01, 0.1, 1.0]$ our proposed algorithms S-$\alpha$S-MDAV and D-$\alpha$S-MDAV lead to less information loss for smaller values of $\alpha$ for all values of $k$ in CENSUS, EIA and Tarragona datasets, while information loss is increased with $\alpha$. This is because the sensitivity $\Delta(f \circ \mathcal{M})$ is smaller when smaller $\alpha$ is used in S-$\alpha$S-MDAV and D-$\alpha$S-MDAV for microaggrega-

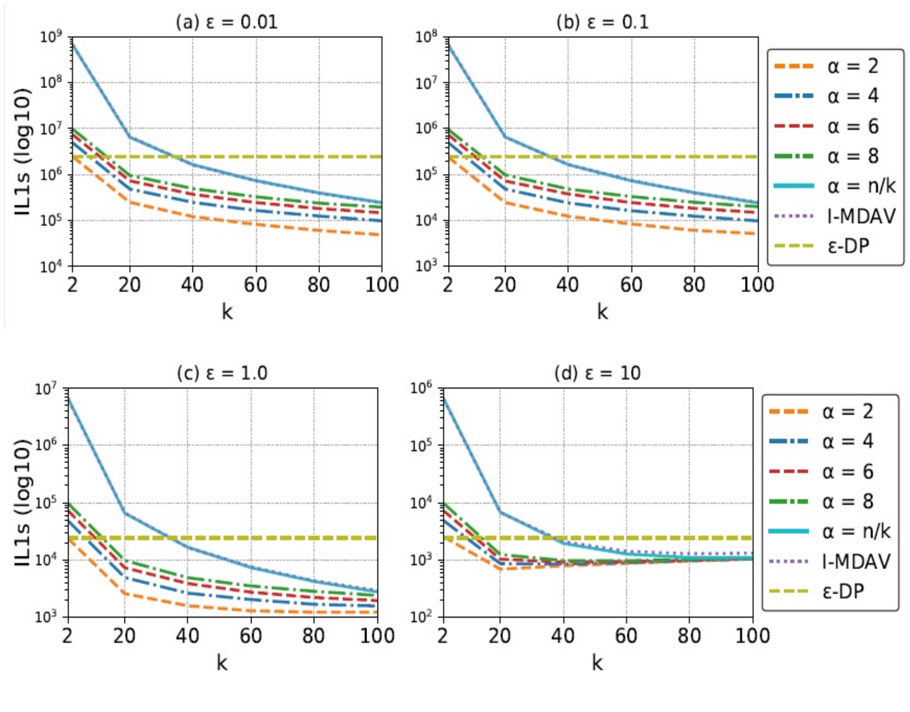


**Figure 5.8:** Comparison on the IL1s of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over CENSUS dataset.

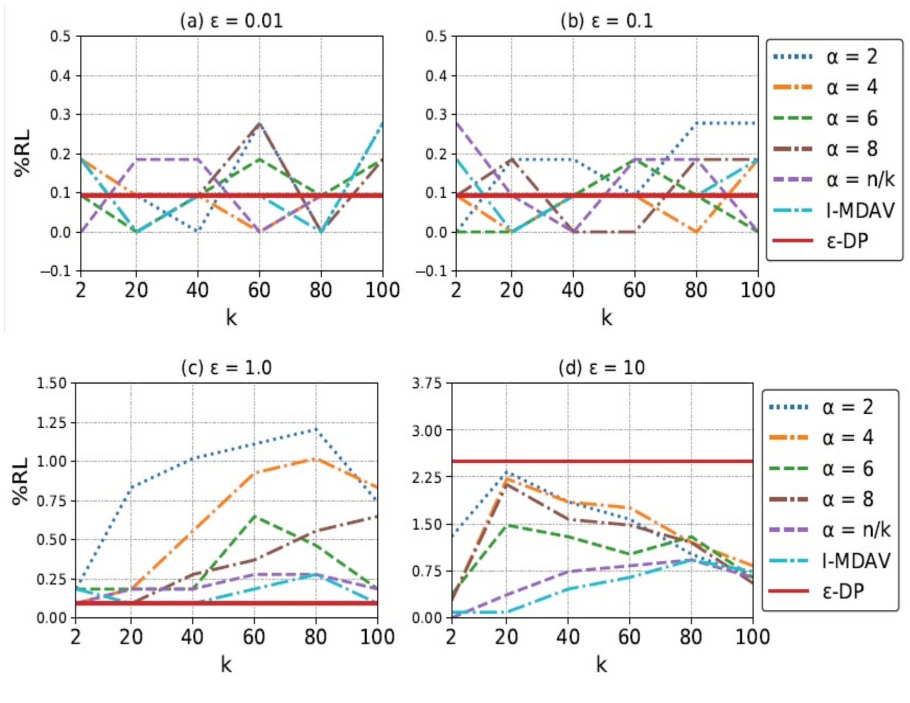


**Figure 5.9:** Comparison on the %RL of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over CENSUS dataset.

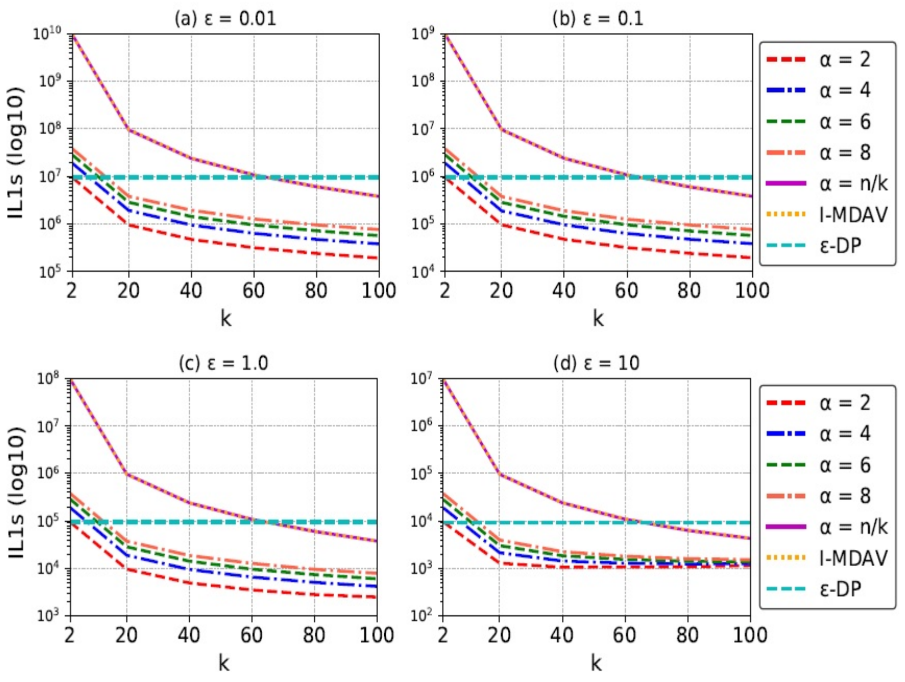


**Figure 5.10:** Comparison on the IL1s of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over EIA dataset.

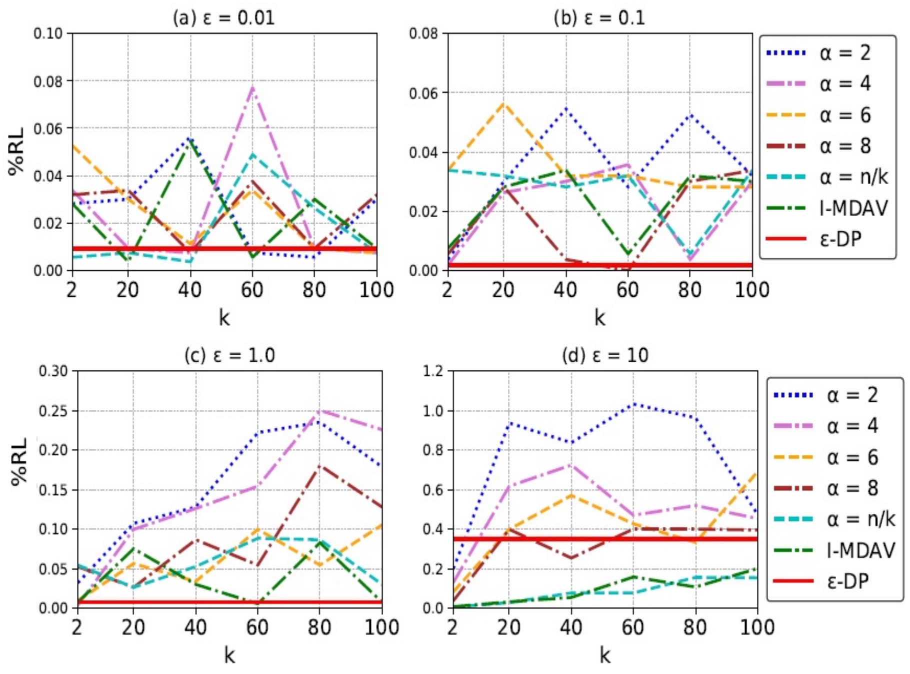


**Figure 5.11:** Comparison on the %RL of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over EIA dataset.

tion. Thus by increasing $\alpha$ more noise is added to differentially private datasets and this results in the increase of information loss. We notice that, for all three datasets, only in one case in Figures 5.8, 5.10 and 5.12, for the highest $\varepsilon$ value ($\varepsilon = 10$), at higher microaggregation level ($k > 70$), the information loss is nearly same by increasing $\alpha$. This is because for the highest $\varepsilon$ value ($\varepsilon = 10$), very little noise addition is required to generate differentially private datasets. As mentioned in Section 5.2, the amount of noise needed to achieve $\varepsilon$-differentially private datasets depends on $\Delta(f)$ and the privacy parameter $\varepsilon$. The amount of information loss directly depends on the value of $\varepsilon$ and $\alpha$ because by increasing $\alpha$ the sensitivity $\Delta(f \circ \mathcal{M})$ increases. The experiments show that our proposed algorithms S-$\alpha$S-MDAV and D-$\alpha$S-MDAV lead to less information loss for every value of $\varepsilon$ as compared to I-MDAV and $\varepsilon$-DP for smaller values of $\alpha$ in all datasets. This is because the sensitivity $\Delta(f \circ \mathcal{M})$ is significantly reduced when a smaller $\alpha$ is used in S-$\alpha$S-MDAV and D-$\alpha$S-MDAV for microaggregation.

I also noticed that by approximating a query $f$ to $f \circ \mathcal{M}$ via microaggregation, the errors caused by random noise that depends on the sensitivity of $f \circ \mathcal{M}$ dominate the impact on the utility of differentially private datasets generated via microaggregation, compared to the errors existing between the original and microaggregated datasets.

**Privacy.** I conducted experiments to compare the percentage of RL of microaggregated datasets that are generated by S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV under varying $k$ between 2 to 100. The results are shown in Tables 5.2, 5.4, and 5.6. I observed that, for these three datasets CENSUS, EIA, and Tarragona, for every value of $\alpha$ and $k$, S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV result in the higher percentage of RL. In order to attain RL below 5%, as highlighted, S-$\alpha$S-MDAV, D-$\alpha$S-MDAV, and I-MDAV require $k \geq 20$. However for $k \geq 20$, I-MDAV yields a slightly lower percentage of RL than S-$\alpha$S-MDAV and D-$\alpha$S-MDAV. This is because the error caused by microaggregation using S-$\alpha$S-MDAV and D-$\alpha$S-MDAV is less as compared with I-MDAV.

Then, to verify the overall privacy of $\varepsilon$-differentially private datasets, I conducted experiments to compare the RL between the original and $\varepsilon$-differentially private datasets generated using our algorithms S-$\alpha$S-MDAV, D-$\alpha$S-MDAV and the baseline methods I-MDAV and $\varepsilon$-DP. The results obtained by S-$\alpha$S-MDAV and D-$\alpha$S-MDAV are quite similar, thus for the purpose of clear visualization we only presented the results obtained using D-$\alpha$S-MDAV. Figures 5.9, 5.11 and 5.13 present our experimental results. The percentage of RL for $\varepsilon$-DP is displayed as horizontal lines, as $\varepsilon$-DP does not depend on $k$. I notice that, for the highest $\varepsilon$ value ($\varepsilon = 10$), in smaller datasets CENSUS and Tarragona, as shown in Figures 5.9 and 5.13, the percentage of RL is lower with S-$\alpha$S-MDAV, D-$\alpha$S-MDAV and I-MDAV as compare to $\varepsilon$-DP. Nevertheless, for larger dataset EIA in Figure 5.11 with the highest $\varepsilon$ value ($\varepsilon = 10$), S-$\alpha$S-MDAV and D-$\alpha$S-MDAV yield less percentage of RL for higher values of $\alpha$. However I-MDAV yields slightly lower percentage of RL than S-$\alpha$S-MDAV and D-$\alpha$S-MDAV. This is because the error caused by microaggregation and differential privacy using

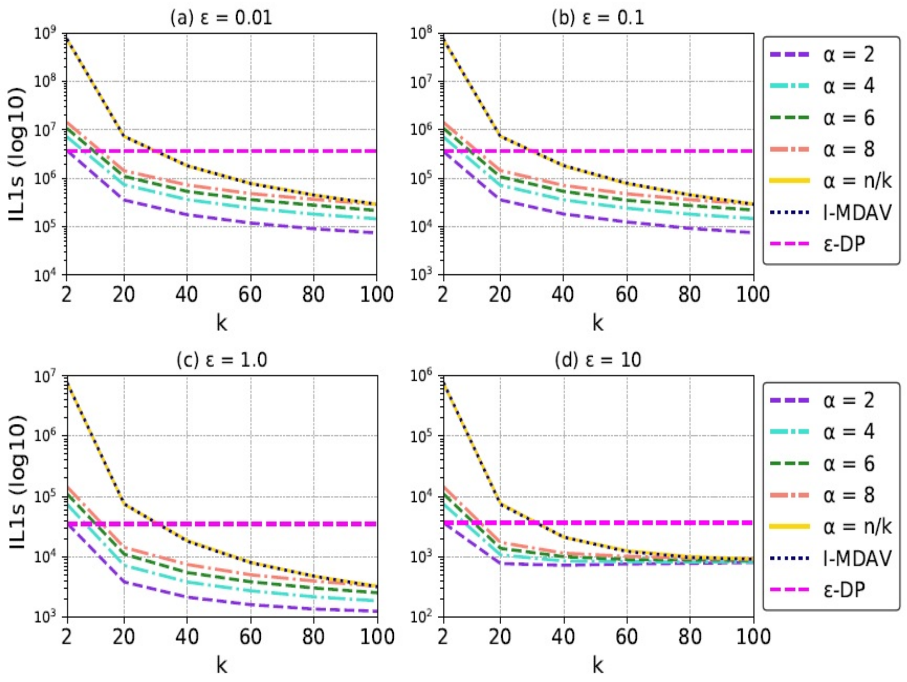


**Figure 5.12:** Comparison on the IL1s of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over Tarragona dataset.

S-$\alpha$S-MDAV and D-$\alpha$S-MDAV is less as compared with I-MDAV and thus overall less noise is being added to datasets for larger $\varepsilon$ and smaller $\alpha$. For values $\varepsilon = [0.01, 0.1]$, in CENSUS and Tarragona datasets the percentages of RL of S-$\alpha$S-MDAV and D-$\alpha$S-MDAV are very similar with I-MDAV and $\varepsilon$-DP and hardly vary when $k$ and $\alpha$ increase. For such low $\varepsilon$-values, the percentage of RL stays around 0.1%. However, for larger dataset EIA, the percentages of RL of S-$\alpha$S-MDAV and D-$\alpha$S-MDAV are very similar with I-MDAV but higher as compared to $\varepsilon$-DP. This is because for such smaller values $\varepsilon = [0.01, 0.1]$ more noise is being added to datasets with higher $\alpha$. For values $\varepsilon = 1.0$, in CENSUS and EIA, the percentage of RL for smaller values of $\alpha$ is higher with S-$\alpha$S-MDAV and D-$\alpha$S-MDAV as compared to I-MDAV and $\varepsilon$-DP. This is because for $\varepsilon = 1.0$ less noise is being added to datasets with smaller $\alpha$. Nevertheless, in the smaller dataset Tarragona the percentages of RL of S-$\alpha$S-MDAV and D-$\alpha$S-MDAV are similar with I-MDAV but smaller as compared to $\varepsilon$-DP for different values of $k$ and $\alpha$.

**Utility Vs Privacy Discussion.** To enhance the utility of differentially private datasets an original query $f$ is approximated to $f \circ \mathcal{M}$ since $f$ is applied to a microaggregated dataset rather than the original dataset. This thus introduces two kinds of errors: one is the random noise to guarantee $\varepsilon$-differential privacy, and the other one is due to computing $f$ over the microaggregated dataset instead of the original dataset. We have noticed that S-$\alpha$S-MDAV and D-$\alpha$S-MDAV outperform I-MDAV in reducing the second kind of error by generating more homogeneous clusters during microaggre-

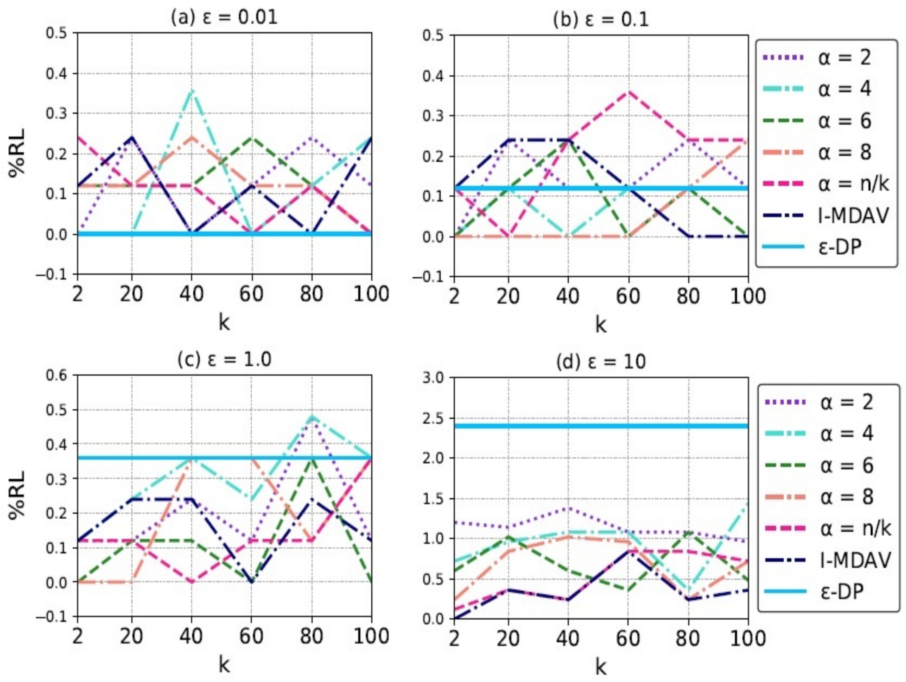


**Figure 5.13:** Comparison on the %RL of D-$\alpha$S-MDAV, and I-MDAV under varying $k$, $\alpha$ and $\varepsilon$ over Tarragona dataset.

gation. Thus the error introduced during the microaggregation process have reduced significantly which enhance the overall utility of the published data. Additionally, S-$\alpha$S-MDAV and D-$\alpha$S-MDAV also guarantee privacy by providing RL less than 5% for $k \geq 20$. This shows that both proposed algorithms preserve privacy while enhancing overall utility of the published data. On the other hand, for the first kind of error which depends on sensitivity, S-$\alpha$S-MDAV and D-$\alpha$S-MDAV only provide better utility when $\alpha$ is small because sensitivity depends on the value of $\alpha$. As we have mentioned that the sensitivity for D-$\alpha$S-MDAV is $(\alpha \times \Delta(f)/k)$, thus to achieve $(\alpha \times \Delta(f)/k) \leq \Delta(f)$, D-$\alpha$S-MDAV requires $k \geq \alpha$. With a small $\alpha$, a less amount of noise is being added to differentially private datasets. Therefore, by approximating a query $f$ to $f \circ \mathcal{M}$ via microaggregation, the errors caused by random noise that depends on the sensitivity of $f \circ \mathcal{M}$ dominate the impact on the utility of differentially private datasets generated via microaggregation, compared to the errors existing between the original and microaggregated datasets. However, by reducing sensitivity we can increase the utility of a dataset but compromising privacy, as we achieve a higher percentage of RL with less sensitivity. Hence, there exists a trade-off between utility and privacy. Adding more noise provides better privacy but less utility; conversely, adding less noise yields better utility but less privacy.

In the experiments, the first kind of error is much larger than the second kind of error in terms of the information loss in $\varepsilon$-differentially private datasets. Hence, by reducing sensitivity we can increase the utility of $\varepsilon$-differentially private datasets without compromising privacy.

## 5.7 2-stable Vs $\alpha$-stable Microaggregation

By using 2-stable microaggregation the amount of noise added into differentially private datasets can be considerably reduced, regardless of the size of a dataset. However, 2-stable microaggregation still has some limitations: 1) the cluster correspondence in microaggregated datasets is restricted, i.e., at most two pairs of corresponding clusters differ in a single record; 2) unable to achieve the insensitive property. The first limitation leads to limit the error reduction during microaggregation process, and due to the later, 2-stable microaggregation does not guarantee differential privacy if we go beyond a specific pair of neighboring datasets. On the other hand, $\alpha$-stable microaggregation extends 2-stable microaggregation by introducing a general parameter $\alpha$. In doing so, 2-stable microaggregation becomes a special case of $\alpha$-stable microaggregation with $\alpha = 2$, where $\alpha$-stable microaggregation enforces that at most $\alpha$ pairs of corresponding clusters in microaggregated neighboring datasets can differ in a single record. Thus, the amount of noise added to differentially private datasets can always be controlled by the parameter $\alpha$, and if the value of $\alpha = 2$ it corresponds to 2-stable microaggregation. Conceptually, $\alpha$ indicates the trade-off between errors introduced during microaggregation and differential privacy. However, to guarantee differential privacy $\alpha$-stable microaggregation requires a partial ordering $\preceq_p$ over the elements in $X \sim Y$, where 2-stable microaggregation does not have such property, thus 2-stable microaggregation does not guarantee differential privacy if we go beyond a specific pair of neighboring datasets.

## 5.8 Summary

In this chapter, I have introduced a novel microaggregation-based framework to generate $\varepsilon$-differentially private datasets, based on the notion *$\alpha$-stable microaggregation* to characterize the correspondence of clusters in microaggregated datasets. I have also proposed two algorithms *sequential $\alpha$-stable microaggregation* and *decisional $\alpha$-stable microaggregation* that can ensure the correspondence of clusters in the microaggregated datasets of two neighboring datasets. Both algorithms aimed to provide privacy to individuals' data while preserving data utility by overcoming the limitations of state-of-the-art methods. I have experimentally verified the effectiveness of the algorithms as compared to the baseline approaches.

# Part II

# Graph Data Publishing

Chapter 6

# dK-Microaggregation: Publishing Graphs with Edge Differential Privacy

## 6.1 Overview

In this chapter, I study the problem of publishing anonymized graphs under the guarantee of edge differential privacy (edge-DP). I observe that the $dK$-graph model [71; 72] for analyzing network topologies can serve as a good basis for generating structure-preserving anonymized graphs. Essentially, the $dK$-graph model generates $dK$-graphs by retaining a series of network topology properties being extracted from $d$-sized subgraphs in an original graph. In order to reduce the amount of random noise without compromising $\varepsilon$-differential privacy, I incorporate microaggregation techniques [21] into the dK graph model to reduce the sensitivity of queries. This enables us to apply perturbation on network topology properties at a flexible level of granularity, depending on the degree of microaggregation.

Figure 6.1 provides a high-level overview of the proposed framework called *dK-Microaggregation*. Given two neighboring graphs $G \sim G'$, network topology properties such as $dK$-distributions [72] are first extracted from each graph. Then a $dK$-distribution goes through a microaggregation procedure, which consists of partition and aggregation. After that, the microaggregated $dK$-distribution is perturbed, yielding a $\varepsilon$-differentially private $dK$-distribution. Finally, based on the perturbed $d$K-distribution, $\varepsilon$-differentially private $dK$-graphs are generated. That is, for two neighboring graphs $G \sim G'$, their corresponding anonymized graphs generated by this framework are $\varepsilon$-indistinguishable.

The main contributions of this chapter are as follows:

- I present a novel framework, called *dK-microaggregation*, that can leverage a series of network topology properties to generate $\varepsilon$-differentially private anonymized graphs.
- I propose a distance constrained algorithm for approximating dK-distributions

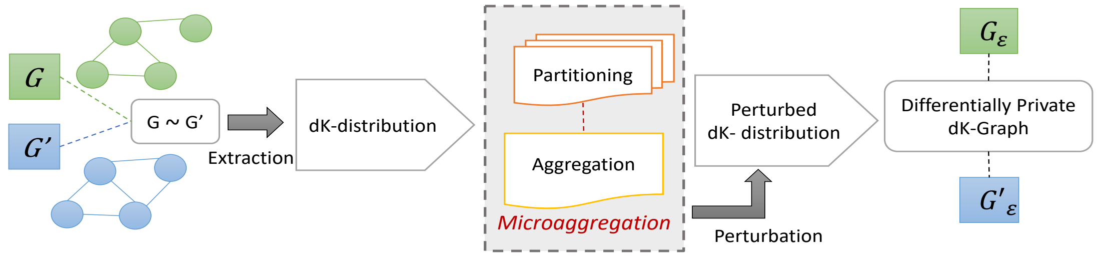


**Figure 6.1**: A high-level overview of *dK-Microaggregation*.

of a graph via microaggregation within the proposed framework, which enables us to reduce the amount of noise being added into $\varepsilon$-deferentially private anonymized graphs.

- I empirically verify the noise reduction of the proposed framework on three real-world networks. It shows that the framework can effectively enhance the utility of generated anonymized graphs by providing better within cluster homogeneity and reducing the amount of noise, in comparison with the state-of-the-art methods.

The rest of this chapter is organized as follows. Section 6.2 introduces the problem definition. Section 6.3 presents the framework *dK-Microaggregation* for generating $\varepsilon$-differentially private $dK$-graphs. Section 6.4 discusses algorithms for microaggregating $dK$-distributions. Section 6.5 theoretically shows that $dK$-graphs generated in the proposed framework are differentially private. Section 6.6 discusses the experimental results, which empirically verify the noise reduction and privacy guarantee of the proposed algorithm against the baseline algorithms. Section 6.7 summarises the chapter.

## 6.2 Problem Definition

Let $G = (V, E)$ be a simple undirected graph, where $V$ is the set of nodes and $E$ the set of edges in $G$, $deg(v)$ denote the degree of a node $v$, and $deg(G)$ denote the maximum degree of $G$.

**Definition 16.** (Edge neighboring graphs) *Two graphs $G = (V, E)$ and $G' = (V', E')$ are said to be* edge neighboring graphs, *denoted as $G \overset{e}{\sim} G'$, iff $V = V'$, $E \subset E'$ and $|E| + 1 = |E'|$.*

The $dK$-graph model [72] offers a systematized method to extract subgraph degree distributions from a given graph, i.e. *dK-distributions*. For $d$ nodes $v_1, \ldots, v_d$ of degrees $g_1, \ldots, g_d$, where $d \in [1, |V|]$, let $T$ be the set of all degree tuples $t = (g_1, \ldots, g_d)$ in a graph $G$.

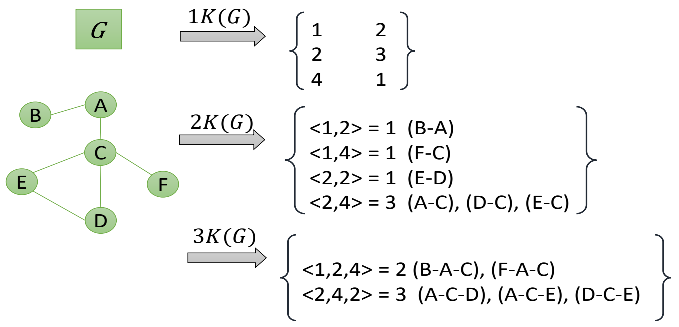

G
1K(G)
1 2
2 3
4 1
2K(G)
<1,2> = 1 (B-A)
<1,4> = 1 (F-C)
<2,2> = 1 (E-D)
<2,4> = 3 (A-C), (D-C), (E-C)
3K(G)
<1,2,4> = 2 (B-A-C), (F-A-C)
<2,4,2> = 3 (A-C-D), (A-C-E), (D-C-E)
A
B
C
D
E
F

**Figure 6.2**: An illustrative example of *dK-distributions*.

**Definition 17.** (dK-DISTRIBUTION) *A* dK-distribution *over a graph $G = (V, E)$, denoted as $dK(G)$, is a probability distribution $p : T \to \mathbb{N}$ defined by $p(t) = m$, where $m$ is the number of the subgraphs with degrees $g_1, \ldots, g_d$.*

For a given graph $G$, $1K$-distribution captures the degree distribution, $2K$-distribution captures the joint degree distribution, i.e. the number of edges between nodes of different degrees, and $3K$-distribution captures the clustering coefficient distribution, i.e. the number of triangles and wedges connecting nodes of different degrees. When $d = |V|$, a $dK$-distribution specifies the entire graph. For larger values of $d$, the $dK$-distributions capture more complex properties of a graph at the expense of higher computational overhead [72]. For simplicity of expression, I do not consider $0k$-distribution (i.e., average degree) in this work.

**Example 6.** *Figure 6.2 illustrates the dK-distributions over a graph G for $d = 1, 2, 3$. When $d = 1$, $1K$-distribution is a degree distribution and I have $p(1) = 2$, $p(2) = 3$ and $p(4) = 1$. For instance, $p(1) = 2$ because there are $2$ nodes (i.e., B, and F) with degree $1$ in G. When $d = 2$, $2K$-distribution is a joint degree distribution and I have $p(1, 2) = 1$, $p(1, 4) = 1$, $p(2, 2) = 1$ and $p(2, 4) = 3$. For instance, $p(2, 4) = 3$ because G contains $3$ edges between $2$ degree nodes (i.e., A, D, and E) and $4$ degree node (i.e., C). Similarly, when $d = 3$, $3K$-distribution is a clustering coefficient distribution and I have $p(1, 2, 4) = 2$ and $p(2, 4, 2) = 3$. For instance, $p(2, 4, 2) = 3$ because G contains $1$ triangle and $2$ wedges formed by three nodes with degrees $2$, $4$, and $2$, respectively. In the same way, when $d = |V|$, dK-distribution is the graph itself, i.e., $p(1, 2, 4, 1, 2, 2) = 1$.*

To describe how a $dK$-distribution is extracted from a graph, we define the notion of $dK$ function.

**Definition 18.** (dK-FUNCTION) *Let $\mathcal{D}$ be the set of all possible dK-distributions over G. A* dK-function *$f^{dK} : \mathcal{G} \to \mathcal{D}$ maps a graph $G \in \mathcal{G}$ to its dK-distribution in $\mathcal{D}$ s.t. $f^{dK}(G) = dK(G)$.*

Following the previous work [72], I define *dK-graph* as a graph that can be constructed through reproducing the corresponding $dK$-distribution.

**Definition 19.** (dK-graph) *A* $dK$-graph *over* $dK(G)$ *is a graph in which connected subgraphs of size* $d$ *satisfy the probability distribution in* $dK(G)$.

Conceptually, a $dK$-graph is considered as an anonymized version of an original graph $G$ that retains certain topological properties of $G$ at a chosen level of granularity. I aim to generate $dK$-graphs with $\varepsilon$-differential privacy guarantee for preserving privacy of structural information between nodes of a graph (edge differential privacy). I formally define *differentially private dK-graph* below.

**Definition 20.** (Differentially private dK-graphs) *A randomized mechanism* $\mathcal{K}$ *provides* $\varepsilon$*-differentially private dK-graphs, if for each pair of edge neighboring graphs* $G \overset{e}{\sim} G'$ *and all possible outputs* $\mathcal{G}^{dK} \subseteq range(\mathcal{K})$*, the following holds*

$$Pr[\mathcal{K}(G) \in \mathcal{G}^{dK}] \leq e^{\varepsilon} \times Pr[\mathcal{K}(G') \in \mathcal{G}^{dK}]. \tag{6.1}$$

$\mathcal{G}^{dK}$ is a family of dK-graphs, and $\varepsilon > 0$ is the *differential privacy parameter*. Smaller values of $\varepsilon$ provide stronger privacy guarantees [29].

## 6.3 dK-Microaggregation Framework

In this section, I present the framework *dK-Microaggregation* for generating $\varepsilon$-differentially private $dK$-graphs. Without loss of generality, I will use $2K$-distribution to illustrate this proposed framework. This is due to two reasons:

1. As previously discussed in [71; 72], the $d = 2$ case is sufficient for most practical purposes.
2. $dK$-generators for $d = 2$ have been well studied [71; 38], whereas $dK$-generators for $d \geq 3$ have not been yet discovered [38].

Previous studies [85; 97] have shown that, changing a single edge in a graph may result in one or more changes on tuples in its corresponding dK-distribution. The following lemma states the maximum number of changes between the $2K$-distributions of two edge neighboring graphs.

**Lemma 3.** *Let* $G \overset{e}{\sim} G'$ *be two edge neighboring graphs. Then* $f^{dK}(G)$ *and* $f^{dK}(G')$ *differ in at most* $4 \times h + 1$ *tuples (entries), where* $d = 2$ *and* $h = max(\{deg(G), deg(G')\})$.

In this work, for each dK-distribution $D$, I want to generate $D_\varepsilon$ that is an anonymized version of $D$ satisfying $\varepsilon$-differential privacy. Thus, I view the response to a dK function $f^{dK}$ for $d = 2$ as a collection of responses to *degree queries*, one for each tuple (entry) in a $2K$ distribution.

**Definition 21.** (Degree query) *A* degree query $f_q : f^{dK}(G) \rightarrow \mathbb{N}$ *maps a degree tuple* $t \in f^{dK}(G)$ *to a frequency value in* $\mathbb{N}$ *s.t.* $(t, f_q(G)) \in f^{dK}(G)$.

To guarantee $\varepsilon$-differential privacy for each $f_q$, I can add random noise into the real response $f_q(G)$, yielding a randomized response $f_q(G) + Lap(\Delta(f_q)/\varepsilon)$, where $\Delta(f_q)$ denotes the sensitivity of $f_q$ and $Lap(\Delta(f_q)/\varepsilon)$ denotes random noise drawn from a Laplace distribution.

Given a set of degree queries $\{f_q\}$, if the response to each $f_q$ satisfies $\varepsilon$-differential privacy, by the parallel composition property of differential privacy [76], I can generate $D_\varepsilon$ that satisfies $\varepsilon$-differential privacy. However, the total amount of random noise being added into the responses can be very high, particularly when a graph is large. To control the amount of random noise and thus increase the utility of $D_\varepsilon$, I microaggregate similar tuples (entries) in $D$ before adding noise. Thus, the $dK$-function $f^{dK}$ is replaced by $f^{dK} \circ \mathcal{M}$, i.e., I run $f^{dK}$ on the microaggregated $dK$-distribution $\overline{D}$ resulting from running a microaggregation algorithm $\mathcal{M}$ over $D$. The response to $f^{dK} \circ \mathcal{M}$ is a collection of responses to *microaggregated degree queries*, one for each cluster in $\overline{D}$.

**Definition 22.** (Microaggregated degree query) *A microaggregate degree query $f_q^* : f^{dK}(G) \rightarrow \mathbb{N}$ maps a set of degree tuples $T$ in $f^{dK}(G)$ to a frequency value in $\mathbb{N}$ s.t. $f_q^*(G) = sum(\{f_q(G) | t = (g_1, g_2), t \in T, (t, f_q(G)) \in f^{dK}(G)\})$.*

Indeed, I can see that $f_q$ is a special case of $f_q^*$ since $f_q(G) = f_q^*(G)$ holds for $T = \{t\}$. By Lemma 3, I have the following lemma about $f_q$ and $f_q^*$.

**Lemma 4.** *The sensitivity of both $f_q$ and $f_q^*$ on a graph $G$ is upper bounded by $(4 \times deg(G) + 1)$.*

For each cluster in $\overline{D}$ that is resulted from running $\mathcal{M}$, only one aggregated frequency value for a cluster of tuples is returned by a microaggregated degree query. Thus, $f^{dK} \circ \mathcal{M}$ is less "sensitive" when the number of clusters in $\overline{D}$ is smaller. By Lemma 4 and the fact that changing one edge on a graph may lead to changes on multiple clusters in $\overline{D}$, I have the following lemma about the sensitivity of $f^{dK} \circ \mathcal{M}$.

**Lemma 5.** *Let $C_1, \ldots, C_z$ be the clusters in $\overline{D}$ resulting from running $\mathcal{M}$ over $f^{dK}(G)$. Then the sensitivity of $f^{dK} \circ \mathcal{M}$ is upper bounded by $(4 \times h + 1) \times z$, where $z$ is the number of clusters and $h = max(\{deg(G), deg(G')\})$.*

Generally, $dK$-microaggregation works in the following steps. First, it extracts a dK-distribution from a graph. Then, it microaggregates the dK-distribution and perturbs the microaggregated dK-distribution to generate $\varepsilon$-differentially private dK-distribution. Finally, a dK-graph is generated.

## 6.4 Proposed dK-Microaggregation Algorithms

In this section, I discuss algorithms for microaggregating dK-distributions. Generally, a microaggregation algorithm for dK-distributions $\mathcal{M} = (\mathcal{M}_p, \mathcal{M}_a)$ consists of two phases: (a) *Partition* - similar tuples in a dK-distribution are partitioned into the same cluster; (b) *Aggregation* - the frequency values of tuples in the same cluster

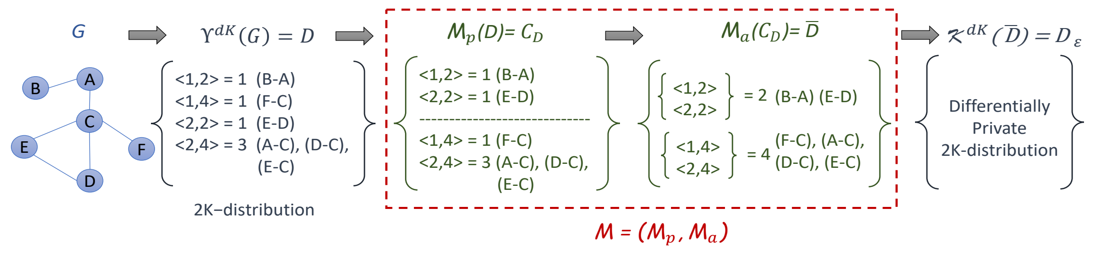

G
$\Upsilon^{dK}(G) = D$
<1,2> = 1 (B-A)
<1,4> = 1 (F-C)
<2,2> = 1 (E-D)
<2,4> = 3 (A-C), (D-C), (E-C)
2K–distribution
$M_p(D) = C_D$
$M_a(C_D) = \overline{D}$
<1,2> <2,2> = 2 (B-A) (E-D)
<1,4> <2,4> = 4 (F-C), (A-C), (D-C), (E-C)
$M = (M_p, M_a)$
Differentially Private 2K-distribution

**Figure 6.3**: An illustration of the proposed dK-Microaggregation algorithms.

are aggregated. As illustrated in Figure 6.3, a 2$K$-distribution $D$ is partitioned into multiple clusters by a partitioning function of a microaggregation algorithm $\mathcal{M}_p$, i.e., $\mathcal{M}_p(D) = \mathcal{C}_D$, where $\mathcal{C}_D = \{C_1, \dots C_z\}$. Then, the frequency values of tuples in each cluster $C_i \in \mathcal{C}_D$ are aggregated by an aggregation function of a microaggregation algorithm $\mathcal{M}_a$, i.e. $\mathcal{M}_a(\mathcal{C}_D) = \overline{D}$.

### 6.4.1 MDAV-dK algorithm.

Given a dK-distribution $D = f^{dK}(G)$ over a graph $G$, a simple way of microaggregating $D$ is to partition $D$ in such a way that each cluster contains at least $k$ tuples. For this, I use a simple microaggregation heuristic, called *Maximum Distance to Average Vector* (MDAV) [21], which can generate clusters of the same size $k$, except one cluster of size between $k$ and $2k - 1$. However, unlike a standard version of MDAV that aggregates each cluster by replacing each tuple in the cluster with a representative tuple, I perform aggregation to aggregate frequency values of tuples in each cluster into an aggregated frequency value. To avoid ambiguity, I call our MDAV-based algorithm for microaggregating $dK$-distributions the *MDAV-dK* algroithm.

### 6.4.2 MPDC-dK algorithm.

For many real-world networks such as Twitter, their degree distributions are often highly skewed. This may lead to highly skewed dK-distributions for such networks. However, due to inherent limitations of MDAV, e.g., the fixed-size constraint, MDAV-dK would suffer significant information loss when evenly partitioning a highly skewed dK-distribution into clusters of the same size. To address this issue, I propose an algorithm called *Maximum Pairwise Distance Constraint* (MPDC-dK).

MPDC-dK aims to partition a dK-distribution into clusters under a distance constraint. Specifically, after partitioning, the distances between the corresponding degrees in any two tuples within a cluster should be no greater than a specified distance interval $\tau$. Take a 2K-distribution $D$ for example. Let $(t_1, m_1)$ and $(t_2, m_2)$ be two tuples in a cluster after applying MPDC-dK on $D$, where $t_1 = (g_1, g_1')$ and $t_2 = (g_2, g_2')$. Then, I say that these two tuples satisfy a distance constraint $\tau$ iff max($|g_1 - g_2|$, $|g_1' - g_2'|$)$\leq \tau$. The clustering problem addressed by MPDC-dK is thus to generate the minimum number of clusters in which every pair of tuples from the same cluster

**Algorithm 4:** *MPDC-dK*

```
Input: $D$: dK-distribution;
       $\tau$: distance interval
Output: $\mathcal{C}_D$: set of clusters
1  $\mathcal{C}_D := \phi$
2  $b\_list := [\,]$
3  foreach $(g, g') \in D$ do
4      foreach $b_i \in covering\_boxes((g, g'), \tau)$ do
5          Add $b_i$ to $b\_list$ (if not exist) and increase the count of $b_i$ by 1 in $b\_list$.
6          Add $(g, g')$ to $b_i$ in $b\_list$
7  while $b\_list$ is not empty do
8      $b_{max} \leftarrow$ the box with the maximum count
9      $C_i \leftarrow$ Generate a cluster of degree pairs in $b_{max}$
10     $\mathcal{C}_D := \mathcal{C}_D \cup \{C_i\}$
11     Remove $b_{max}$ from $b\_list$.
12     foreach $(g, g') \in d_{max}$ do
13         foreach $b_i \in covering\_boxes((g, g'), \tau)$ do
14             Remove $(g, g')$ from $b_i$ in $b\_list$
15             Decrease the count of $b_i$ in $b\_list$ by 1 and remove $b_i$ if its count is 0
16 Return $\mathcal{C}_D$
```

satisfies such a distance constraint $\tau$.

The conceptual ideas behind the MPDC-dK algorithm design is to consider degree pair $(g, g')$ of each degree tuple $t = (g, g')$ as coordinates in a two dimensional space, and also treat the above distance constraint $\tau$ as a $\tau$-by-$\tau$ box, denoted by $((x, x'), \tau)$ and centered at $(x, x')$, in the same two dimensional space. Clearly, such a box corresponds to a cluster that satisfies the distance constraint $\tau$, and a box $((x, x'), \tau)$ covers a degree tuple $(g, g')$ iff $x - \tau/2 \leq g \leq x + \tau/2$ and $x' - \tau/2 \leq g' \leq x' + \tau/2$. MPDC-dK employs a greedy algorithm to find the minimum number of boxes (i.e., clusters) that cover all degree pairs as follow:

- MDPC-dK first enumerates all boxes that cover at least one degree pair and records the corresponding counts as the number of degree pairs being covered by these boxes.

- Then, MDPC-dK recursively selects a box with the maximum count (i.e., covering the maximum number of degree pairs) in a greedy manner, assigns these degree pairs in a new cluster, and removes them from other boxes until all boxes are empty.

- After that, MDPC-dK performs aggregation to aggregate the frequency values of tuples in each cluster into an aggregated frequency value.

Algorithm 4 describes the details of our MPDC-dK algorithm. Given a dK-distribution $D$, I start with initializing an empty cluster list $\mathcal{C}_D$ to hold clusters (Line 1) and a list *b_list* to record each box and its corresponding degree pairs, and the total number of degree pairs covered by the box (Line 2). For each degree pair $(g, g')$ of each degree tuple $t = (g, g')$ in $D$, I enumerate boxes that cover $(g, g')$ using a function *covering_boxes* that satisfies the distance constraint $\tau$ (Line 4). For each enumerated box $b_i$ I update the list by adding $(g, g')$ to $b_i$ and increment the count of $b_i$ by 1 (Lines 5-6). After creating *b_list*, I iteratively select a box $b_{max}$ with the maximum count for degree pairs (Line 8), then generate a new cluster of degree pairs in $b_{max}$, and add it into the cluster list (Lines 9-10). I further remove $b_{max}$ and all degree pairs in $b_{max}$ from *b_list* and update the counts of affected boxes in *b_list* (Lines 11-15). The algorithm terminates when *b_list* is empty and returns a set of generated clusters $D'$.

## 6.5 Theoretical Discussion

### 6.5.1 Privacy Analysis

Here, I theoretically show that $dK$-graphs generated in our proposed framework are differentially private. Firstly, by Lemma 4 and 5, I can obtain a $\varepsilon$-differentially $dK$-distribution $D_\varepsilon$ by microaggregating a $dK$-distribution and calibrating the amount of random noise according to the sensitivity of microaggregated degree queries. As $D_\varepsilon$ only contains aggregated frequency values for clusters of tuples in a $dK$-distribution, I perform post-processing using a randomized algorithm $\mathcal{K}'$ to randomly select tuples within each cluster of $D_\varepsilon$ until the aggregated frequency value of the cluster is reached. Previously, Dwork and Roth [30] proved that differential privacy is immune to post-processing, i.e., the composition of a randomized algorithm with a differentially private algorithm is differentially private. This leads to the lemma below.

**Lemma 6.** *Let $D_\varepsilon$ be a $\varepsilon$-differentially private dK-distribution and $\mathcal{K}'$ be a randomized algorithm for post-processing $D_\varepsilon$. Then $\mathcal{K}'(D_\varepsilon)$ is also a $\varepsilon$-differentially private dK-distribution.*

Based on $\mathcal{K}'(D_\varepsilon)$, a $dK$-graph can be generated using a $dK$-graph generator [71; 72]. Following Lemma 6, Definition 20, and the proposition of Dwork and Roth [30] on post-processing, I have the following theorem for the framework, which corresponds to a randomized mechanism $\mathcal{K} = f^{dK} \circ \mathcal{M} \circ \mathcal{K}^{dK} \circ \mathcal{K}' \circ \widehat{f}^{dK}$, where $\widehat{f}^{dK} : \mathcal{D} \to \mathcal{G}$ is a $dK$-graph generator.

**Theorem 1.** *$\mathcal{K}$ generates $\varepsilon$-differentially private dK-graphs.*

### 6.5.2 Complexity Analysis

I analyze the time complexity of the algorithms MDAV-dK and MPDC-dK. For MDAV-dK with a constraint on the minimum size $k$ of clusters, given a $dK$-distribution $D$ as input, the complexity of MDAV-dK for clustering is similar to MDAV [21], i.e. $\mathcal{O}(n^2)$, where $n$ is a number of tuples (entries) in $D$. For MPDC-dK with a constraint on the distance interval $\tau$, in order to generate clusters, MPDC-dK needs to perform a sequential search over all degree pairs in $D$. Firstly, MPDC-dK needs to enumerate boxes for all the degree pairs, and each degree pair is covered by at most $(\tau + 1)^2$ boxes (Line 4 of Algorithm 1), hence the cost of enumerating boxes is $\mathcal{O}(\tau^2 n)$ (Line 3-6 of Algorithm 4).

Secondly, MPDC-dK sorts the boxes based on the corresponding degree pairs being covered, and selects and removes the box with the maximum count iteratively. Although it takes $\mathcal{O}(nlogn)$ to sort and greedily select the box with the maximum count for the first iteration, each later iteration only costs $\mathcal{O}(\tau^2 logn)$ (Line 8 of Algorithm 4) because each box overlaps with at most $4\tau^2$ other boxes and removing one box only affects the count of $\mathcal{O}(\tau^2)$ boxes (Lines 11-15 of Algorithm 4). Hence, the cost of selecting and removing boxes is $\mathcal{O}(\tau^2 nlogn)$ (Lines 7-15 of Algorithm 1). The overall complexity of MPDC-dK for clustering is $\mathcal{O}(\tau^2 nlogn)$.

## 6.6 Experiments

I have evaluated the proposed framework to answer the following questions:

- **Q1.** How does dK-microaggregation reduce the amount of noise added into dK-distributions while still providing $\varepsilon$-differential privacy guarantee?
- **Q2.** How does our microaggregation algorithms perform in providing better within cluster homogeneity for dK-distributions?
- **Q3.** What are the trade-offs between utility and privacy when generating differentially private dK-graphs?

### 6.6.1 Experimental Setup

**Datasets.** I used three network datasets in the experiments:

1. *polbooks*[1] contains 105 nodes and 441 edges. It is a network of books about US politics.
2. *ca-GrQc*[1] contains 5,242 nodes and 14,496 edges. It is a network of scientific collaborative networks between authors and papers.
3. *ca-HepTh*[1] contains 9,877 nodes and 25,998 edges. It is also a network of scientific collaborative networks between authors and papers.

**Baseline methods.** In order to evaluate our proposed framework, I considered the following methods:

1. $\varepsilon$-DP, which is a standard $\varepsilon$-differential privacy algorithm in which noise is added using the Laplace mechanism [29].

[1]*polbooks* is available at http://networkrepository.com/polbooks.php; *ca-GrQc* and *ca-HepTh* are available at http://snap.stanford.edu/data/index.html

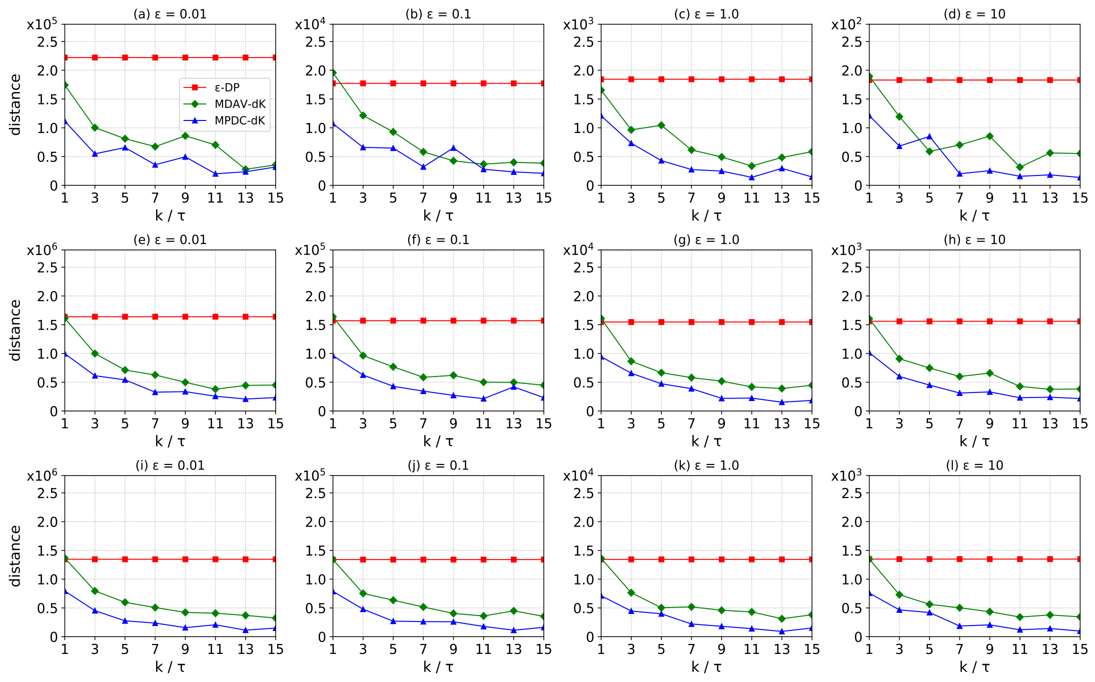


**Figure 6.4:** Comparison on the Euclidean distance between original and perturbed dK-distributions under varying $k$, $\tau$, and $\varepsilon$ over three datasets: (a)-(d) *polbooks* dataset, (e)-(h) *ca-GrQc* dataset, and (i)-(l) *ca-HepTh* dataset.

2. MDAV-dK, which extends the standard microaggregation algorithm MDAV [21] for handling dK-distributions.
3. MPDC-dK is the proposed dK-microaggregation algorithm.

Additionally, I used *Orbis* [71] to generate 2K-distributions.

**Evaluation measures.** I used two evaluation measures:

1. I used Euclidean distance [85] to measure network structural error between original and perturbed dK-distributions.
2. For clustering algorithms, I measure within cluster homogeneity using the sum of absolute error [33] defined as

$$SAE = \sum_{i=1}^{N} \sum_{\forall x_j \in c_i} |x_j - \mu_i| \tag{6.2}$$

where $c_i$ is the set of tuples in cluster $i$ and $\mu_i$ is the mean of cluster $i$.

## 6.6.2 Experimental Results and Discussion

To verify the overall utility of $\varepsilon$-differentially private dK-distribution, I first conducted experiments to compare the structural error between original and perturbed

**Table 6.1**: Performance of MDAV-dK under different values of $k$.

| Datasets | Measures | $k$=1 | $k$=3 | $k$=5 | $k$=7 | $k$=9 | $k$=11 | $k$=13 | $k$=15 |
|---|---|---|---|---|---|---|---|---|---|
| *polbooks* | SAE | 0 | 144.6 | 184.67 | **224.84** | 273.6 | 292.21 | 299.15 | 334.25 |
| | # Clusters | 161 | 53 | 32 | **23** | 17 | 14 | 12 | 10 |
| *ca-GrQc* | SAE | 0 | 1073.3 | 1476 | **1810.5** | 2166.8 | 2313.7 | 2555.5 | 2730 |
| | # Clusters | 1233 | 411 | 246 | **176** | 137 | 112 | 94 | 82 |
| *ca-HepTh* | SAE | 0 | 968.72 | 1304 | 1599.8 | **1893.9** | 2063 | 2232.9 | 2389.7 |
| | # Clusterss | 1295 | 431 | 259 | 185 | **143** | 117 | 99 | 86 |

**Table 6.2**: Performance of MPDC-dK under different values of $\tau$.

| Datasets | Measures | $\tau$=1 | $\tau$=3 | $\tau$=5 | $\tau$=7 | $\tau$=9 | $\tau$=11 | $\tau$=13 | $\tau$=15 |
|---|---|---|---|---|---|---|---|---|---|
| *polbooks* | SAE | 90.72 | **192.15** | 328.96 | 424.2 | 563.73 | 617.63 | 723.06 | 795.77 |
| | # Clusters | 68 | **25** | 13 | 8 | 7 | 5 | 3 | 3 |
| *ca-GrQc* | SAE | 725.38 | **1732.1** | 2630.6 | 3470.6 | 4262.9 | 5176.7 | 6170.1 | 7037.7 |
| | # Clusters | 483 | **178** | 98 | 61 | 42 | 35 | 26 | 20 |
| *ca-HepTh* | SAE | 841.87 | **1761.8** | 2773.3 | 3721.4 | 4719.2 | 5623.8 | 6402.6 | 7034.2 |
| | # Clusters | 412 | **140** | 73 | 37 | 34 | 24 | 19 | 15 |

dK-distributions generated by the algorithms MDAV-dK, MPDC-dK and the baseline method $\varepsilon$-DP. Figure 6.4 presents the experimental results. For $\varepsilon$-DP, I used the following privacy parameters $\varepsilon = [0.01, 0.1, 1.0, 10.0]$, which cover the range of differential privacy levels widely used in the literature [46]. The results for $\varepsilon$-DP is displayed as horizontal lines, as $\varepsilon$-DP does not depend on the parameters $k$ and $\tau$.

From Figure 6.4, it can be seen that, for all three datasets, the proposed algorithms MDAV-dK and MPDC-dK lead to less structural error for every value of $\varepsilon$ as compared to $\varepsilon$-DP. This is because, by approximating a query $f$ to $f \circ \mathcal{M}$ via *dk*-microaggregation, the errors caused by random noise to achieve $\varepsilon$-differential privacy are reduced significantly. Thus, dK-microaggregation introduces overall less noise to achieve differential privacy.

I then conducted experiments to compare the quality of clusters, in terms of within cluster homogeneity, generated by MDAV-dK and MPDC-dK. The results are shown in Tables 6.1 and 6.2. I observe that, for values of $k$ and $\tau$ at which MDAV-dK and MPDC-dK generate almost the same number of clusters, as highlighted in bold, MPDC-dK outperforms MDAV-dK by producing clusters with less SAE over all three datasets. This is consistent with the previous discussion in Section 6.4. As MPDC-dK always partitions degree pairs under a distance constraint rather than a fixed-size constraint, thus it generates more homogeneous clusters as compared to MDAV-dK.

**Discussion.** I have analyzed the trade-offs between utility and privacy of dK-graphs generated in the proposed framework. To enhance the utility of differentially private dK-graphs, I approximated an original query $f$ to $f \circ \mathcal{M}$. This thus introduces two kinds of errors: one is random noise to guarantee $\varepsilon$-differential privacy, and the other one is due to microaggregation. I have noticed that, the second kind of error can be reduced by generating homogeneous clusters during microaggregation. On the other hand, for the first kind of error which depends on the sensitivity of $f \circ \mathcal{M}$, it dominates the impact on the utility of differentially private dK-graphs generated via dk-microaggregation. By reducing sensitivity I can increase the utility of dK-graphs without compromising privacy.

## 6.7 Summary

In this chapter, I have formalized a general microaggregation-based framework for anonymizing graphs that preserves the utility of dK-graphs while enforcing edge differential privacy. Based on the proposed framework, I have proposed an algorithm for microaggregating $dK$-distributions under a distance constraint. I have theoretically analyzed the privacy property of the framework and the complexity of the algorithm. The effectiveness of my work has been empirically verified over three real-world datasets.

Chapter 7

# dK-Projection: Publishing Graph Joint Degree Distribution with Node Differential Privacy

## 7.1 Overview

In this chapter, I study the problem of publishing higher-order network statistics, i.e., *joint degree distribution*, under the guarantee of *node differential privacy* (node-DP). To the best of my knowledge, this is the first study to publish higher-order graph statistics under node-DP, while enhancing graph data utility. I observe that $dK$-distributions [72; 71] can serve as a good basis for representing higher-order network statistics. Informally, $dK$-distributions [72; 71] are a set of reproducible graph properties, which capture degree correlations within $d$-sized subgraphs of a network. As networks are structures of connections between nodes, $dK$-distributions provide rich information about network structures for analysis. To explore the sensitivity of higher-order network statistics under node-DP, I theoretically analyze the sensitivity of $dK$-distributions for $d = 2$, i.e., *joint degree distribution* [73]. It is known that joint degree distribution contains useful information about connectivity in a graph, i.e., given a joint degree distribution, one can always restore both the degree distribution and average degree [73; 23].

To alleviate the challenge posed by high sensitivity of higher-order network statistics under node-DP, I propose a "graph projection" technique that can transform a graph $G$ into a $\theta$-bounded graph $G^\theta$ such that the maximum degree in $G^\theta$ is no larger than a threshold $\theta$. The motivation behind graph projection is to bound the sensitivity of publishing network statistics through a control on node degrees. In doing so, a query $f$ for higher-order network statistics with high sensitivity on a graph $G$ is transformed to an approximate query $f \circ \mathcal{P}$ which has lower sensitivity than $f$, where $\mathcal{P}$ refers to a graph projection algorithm that transforms a graph $G$ to a $\theta$-bounded graph $G^\theta$.

Figure 7.1 provides a high-level overview of the proposed framework. Given a graph $G$, a graph projection algorithm transforms $G$ into a $\theta$-bounded graph $G^\theta$. Then higher-order network statistics such as $dK$-distributions [72] are extracted from

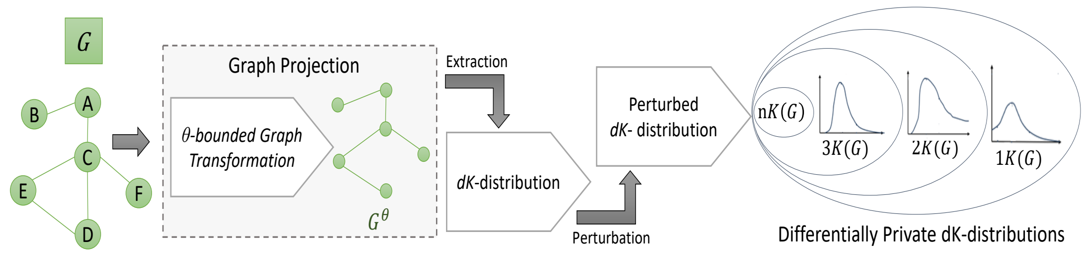

G
A
B
C
E
F
D
Graph Projection
θ-bounded Graph Transformation
$G^\theta$
Extraction
dK-distribution
Perturbed dK- distribution
Perturbation
nK(G)
3K(G)
2K(G)
1K(G)
Differentially Private dK-distributions

**Figure 7.1**: A high-level overview of *dK-Projection*.

$G^\theta$, and extracted $dK$-distributions are perturbed yielding $\varepsilon$-differentially private $dK$-distributions.

The main contributions of this chapter are as follows:

- I present a novel framework to publish higher-order network statistics under node-DP.
- I analyse the sensitivity of publishing *joint degree distribution* in the proposed framework.
- I introduce a new graph projection algorithm to reduce sensitivity of publishing network statistics under node-DP.
- I conduct comprehensive experiments over four real-world networks, and the results demonstrate that the proposed framework can effectively enhance the utility of differentially private network statistics.

The rest of this chapter is organized as follows. Section 7.2 introduces the problem definition. Section 7.3 presents sensitivity analysis to publish joint degree distribution of a graph. Section 7.4 first introduces a novel graph projection technique and then incorporates it into a node-DP releasing mechanism. Section 7.5 discusses the experimental results, which empirically verify the utility enhancement and privacy guarantee of the proposed framework against the baseline methods. Section 7.6 summarises the chapter.

## 7.2 Problem Definition

Let $N_G(v) = \{u \in V | (u,v) \in E\}$ denote the set of neighbors of a vertex $v$ in $G$, $deg(v)$ the degree of a node $v$, and $deg(G) = max\{deg(v)|v \in V\}$ the maximum degree of nodes in $G$. Below, I define the notion of neighboring graphs under node-DP.

**Definition 23.** (Node neighboring graphs) *Two graphs $G = (V,E)$ and $G' = (V',E')$ are said to be* node neighboring graphs, *denoted as $G \overset{n}{\sim} G'$, iff $V' = V \cup \{v^+\}$, $E' = E \cup E^+$, and $E^+$ is the set of all edges incident to $v^+$in $G'$.*

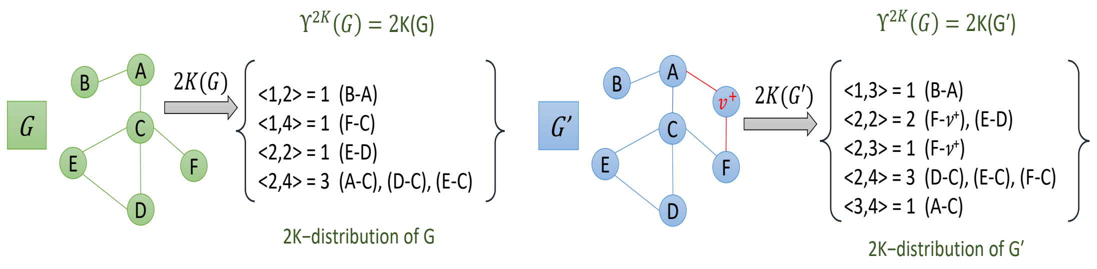


**Figure 7.2:** An illustrative example of *dK-distribution* and its maximum change on two node neighboring graphs $G \overset{n}{\sim} G'$, when $d = 2$.

Given a graph, I represent its topology properties as $dK$-distributions [72]. Recall the definitions of $dK$-distribution and $dK$ function presented in Chapter 6.

**Definition 24.** (DK-DISTRIBUTION) *A* dK-distribution *over a graph $G = (V, E)$, denoted as $dK(G)$, is a probability distribution $p : T \to \mathbb{N}$ such that $p(a_1, \ldots, a_d)$ refers to the total number of connected subgraphs of size $d$ in $G$ with the nodes $\{v_1, \ldots, v_d\}$ and $a_i = deg(v_i)$ for $i = 1, \ldots, d$.*

**Definition 25.** (DK-FUNCTION) *Let $\mathcal{D}$ be the set of all possible dK-distributions over G. A* dK function *$f^{dK} : \mathcal{G} \to \mathcal{D}$ mapping a graph $G \in \mathcal{G}$ to its dK-distribution in $\mathcal{D}$ s.t. $f^{dK}(G) = dK(G)$.*

When $d = 2$, $f^{dK}(G)$ returns the joint degree distribution of $G$, i.e., $p(i, j)$ is a frequency value, referring to the number of edges connecting nodes of degrees $i$ and $j$. Consider Figure 7.2, which depicts the $2K$-distribution of a graph $G$. $p(2, 4) = 3$ because $G$ contains 3 edges between 2 degree nodes (i.e., $A$, $D$, and $E$) and 4 degree node (i.e., $C$).

To release $dK$-distribution under the guarantees of node-DP, I perturb $dK$- distribution by adding controlled noise from Laplace mechanism [29] defined in Chapter 3. Formally, I define the following Laplace mechanism to perturb the $dK$-distribution returned by a $dK$ function over a graph $G$.

**Definition 26.** (PERTURBED DK-DISTRIBUTION) *Let $\varepsilon > 0$ be the* privacy parameter *(smaller values provide stronger privacy guarantees). The following Laplace mechanism is applied to produce a perturbed output of $f^{dK}$:*

$$\mathcal{K}(G) = f^{dK}(G) + Lap\left(\frac{\Delta f}{\varepsilon}\right)^{|V|^d}$$
$$where \quad \Delta f = \max_{G \sim G'}(f^{dK}(G) - f^{dK}(G'))$$
$$and \quad Pr[Lap(\beta) = x] = \frac{1}{2\beta}e^{-|x|/\beta}$$

Here, $\Delta f$ refers to the *sensitivity* of the $dK$-function $f^{dK}$, which is the maximum variation in its output, i.e., $dK$-distribution, over two node neighboring graphs $G \overset{n}{\sim}$

$G'$. We will discuss in detail how to compute the sensitivity of a $dK$-function under node-DP in Section 7.3.

Below, I formulate the notion of $\varepsilon$-differentially private $dK$-distribution (i.e., an anonymized version of $f^{dK}(G)$ satisfying differential privacy) based on node neighboring graphs and perturbed $dK$-distribution.

**Definition 27.** (Differentially Private dK-distribution) *A randomized mechanism $\mathcal{K}$ is $\varepsilon$-differentially private, if for each pair of node neighboring graphs $G \overset{n}{\sim} G'$ and all possible perturbed dK-distributions $\mathcal{D}_\varepsilon \subseteq range(\mathcal{K})$, the following holds:*

$$Pr[\mathcal{K}(G) \in \mathcal{D}_\varepsilon] \leq e^\varepsilon \times Pr[\mathcal{K}(G') \in \mathcal{D}_\varepsilon]. \tag{7.1}$$

The challenge of releasing differentially private $dK$-distributions is to determine how much noise should be added to perturb $dK$-distributions. Adding more noise can better guarantee node-DP; however, data utility deteriorates. When $\varepsilon$ is specified, the magnitude of noise depends on the sensitivity of $dK$-function.

## 7.3 Sensitivity Analysis

In this section, I analyze the sensitivity of $dK$-function $f^{dK}(G)$ for $d = 2$, to publish joint degree distribution of a graph $G$. The goal is to derive the minimum amount of noise needed to achieve node-DP.

Suppose that a node $v^+$ is added to $G$ with a set $E^+$ of edges, each edge $(v^+, v_i) \in E^+$ may cause at most $2 \times deg(G) + 1$ entries of $f^{2K}(G)$ being changed. Thus, for each $v_j \in N(v_i)$, $p(deg(v_i), deg(v_j))$ may decrease by one and $p(deg(v_i) + 1, deg(v_j))$ may increase by one, which amount to the number $2 \times deg(G)$ of entries being changed if each $v_j \in N(v_i)$ has the maximum degree, i.e., $deg(v_j) = deg(G)$. In addition to this, $p(deg(v^+), deg(v_i))$ increases by at least one. Thus, the total number of entries of $f^{2K}(G)$ being changed by all edges in $E^+$ is upper bounded by $(2 \times deg(G) + 1) \times |E^+|$.

**Lemma 7.** *Let $G \overset{n}{\sim} G'$ be two node neighboring graphs. When $d = 2$, $f^{dK}(G)$ and $f^{dK}(G')$ differ in at most $(2 \times deg(G) + 1) \times |E^+|$ entries, where $|E^+|$ is the set of all edges incident to $v^+$.*

Prior studies [59; 85] have shown that, in large social networks, $deg(G)$ is upper bounded by $O(\sqrt{|V|})$. Thus, for such networks, the sensitivity of $2K$-function is upper bounded by $O(2 \times |V|\sqrt{|V|} + |V|)$.

Consider Figure 7.2 in which a node $v^+$ and two edges $\{(v^+, F), (v^+, A)\}$ are added into a graph $G$, resulting in the graph $G'$. There are 7 entries of $f^{2K}(G)$ being changed: (1) $(v^+, F)$ leads to changing 3 entries, i.e., $p(1,4)$ decrease by one, and $p(2,4)$ and $p(2,2)$ increase by one; (2) $(v^+, A)$ leads to changing 5 entries, i.e., $p(1,2)$ and $p(2,4)$ decrease by one, and $p(1,3)$, $p(3,4)$ and $p(2,3)$ increase by one. Although $p(2,4) = 3$ in both $G$ and $G'$, it is changed twice by increasing one and decreasing one. If $G$ is a complete graph with $deg(G) = 4$ and $|E^+| = 2$, 18 entries of $f^{2K}(G)$ would be changed, which is the worst case.

**Algorithm 5:** Stable-Edge-Removal (SER)

```
Input: A graph $G = (V, E)$;
       a projection parameter $\theta$;
       a stable ordering $\Gamma$
Output: A $\theta$-bounded graph $G^\theta = (V, E^\theta)$
1 $E^\theta := E$
2 $deg[v] \leftarrow deg(v), \forall v \in V$
3 foreach $(v, u) \in Seq(E, \succ_\Gamma)$ do
4     if $deg[v] > \theta$ then
5         $E^\theta \leftarrow E^\theta \setminus \{(u, v)\}$
6         $deg[u] \leftarrow deg[u] - 1$
7         $deg[v] \leftarrow deg[v] - 1$
8 Return $G^\theta = (V, E^\theta)$
```

## 7.4 Proposed dK-Projection Approach

In this section, I first introduce a novel graph projection technique and then incorporate it into a node-DP releasing mechanism.

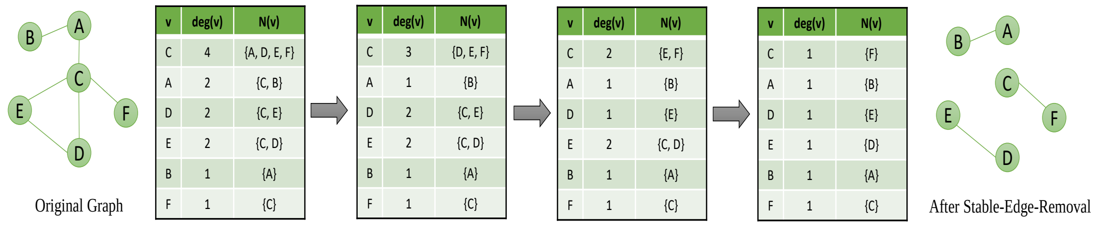
### 7.4.1 Stable-Edge-Removal Graph Projection

Existing graph projection techniques generally fall into two categories: (1) vertex ordering methods such as truncation [53]; (2) edge ordering methods such as edge-removal [5] and edge-addition [14]. However, these methods have some limitations. Vertex ordering methods often truncate all nodes $v \in V$ with $deg(v) > \theta$ [53], which severely affects the topological structure of a graph. Indeed, up to $\theta$ edges incident to these nodes may be preserved in a $\theta$-bounded graph. For edge ordering methods [5; 14], they handle edges based on a random edge ordering, which may cause an excessive number of edges to be lost from an original graph.

To alleviate these limitations, I propose *Stable-Edge-Removal* (SER) that transform a graph $G$ to a $\theta$-bounded graph $G^\theta$ with $\theta < deg(G)$ based on a two-level ordering strategy on $G$.

**Definition 28.** *A* two-level ordering *over $G = (V, E)$ is a pair $\Gamma = (\succ_N, \succ_V)$ where $\succ_N$ is a* local neighbor ordering *such that, for each $v \in V$, there is a bijection: $N_G(v) \to \{1, \ldots, |N_G(v)|\}$; $\succ_V$ is a* global node ordering *such that there is a bijection: $V \to \{1, \ldots, |V|\}$.*

The intuition behind such a two-level ordering is to provide the flexibility of ranking nodes from two aspects: (i) global importance in a graph, and (ii) local importance in neighbourhoods. Given a two-level ordering $\Gamma$, an edge ordering is defined. Specifically, for a graph $G = (V, E)$, there exists a total ordering $\succ_\Gamma$ on edges in $E$ such that, for any two edges $e_1 = (v_1, u_1)$ and $e_2 = (v_2, u_2)$ (assume



| v | deg(v) | N(v) |
|---|---|---|
| C | 4 | {A, D, E, F} |
| A | 2 | {C, B} |
| D | 2 | {C, E} |
| E | 2 | {C, D} |
| B | 1 | {A} |
| F | 1 | {C} |

| v | deg(v) | N(v) |
|---|---|---|
| C | 3 | {D, E, F} |
| A | 1 | {B} |
| D | 2 | {C, E} |
| E | 2 | {C, D} |
| B | 1 | {A} |
| F | 1 | {C} |

| v | deg(v) | N(v) |
|---|---|---|
| C | 2 | {E, F} |
| A | 1 | {B} |
| D | 1 | {E} |
| E | 2 | {C, D} |
| B | 1 | {A} |
| F | 1 | {C} |

| v | deg(v) | N(v) |
|---|---|---|
| C | 1 | {F} |
| A | 1 | {B} |
| D | 1 | {E} |
| E | 1 | {D} |
| B | 1 | {A} |
| F | 1 | {C} |

**Figure 7.3**: An illustration of Stable-Edge-Removal with $\theta = 1$.

$v_i \succ_V u_i$ for simplicity), I have $e_1 \succ_\Gamma e_2$ if and only if $v_1 \succ_V v_2$, or $u_1 \succ_N u_2$ when $v_1 = v_2$.

For two node neighboring graphs $G \overset{n}{\sim} G'$, one important property of a graph projection algorithm $\mathcal{P}$ is to ensure that $\theta$-bounded graphs $\mathcal{P}(G)$ and $\mathcal{P}(G')$ are also node neighboring graphs, i.e., $\mathcal{P}(G) \overset{n}{\sim} \mathcal{P}(G')$. To obtain this property, I require $\Gamma$, upon which $\mathcal{P}$ depends, to be stable on two node neighboring graphs $G \overset{n}{\sim} G'$.

**Definition 29.** (Stable Ordering) *Given two node neighboring graphs $G = (V, E)$ and $G' = (V', E')$, a two-level ordering $\Gamma = (\succ_N, \succ_V)$ is* stable *if and only if,*

- *for any node $v$ in $V \cap V'$, the relative orderings of their common neighbors in $(N_G(v) \cap N_{G'}(v))$ are the same in $\succ_N (G)$ and $\succ_N (G')$, and*
- *for any two nodes in $V \cap V'$, their relative orderings are the same in $\succ_V (G)$ and $\succ_V (G')$.*

Algorithm 5 describes the main steps of the *SER* algorithm. Given an input graph $G = (V, E)$, a projection parameter $\theta$, and a stable ordering $\Gamma$ as describe above, I initialize $E^\theta$ with all edges in $E$ and a list *deg* which holds degrees of all nodes in $V$, where $V$ is the set of all nodes (Line 1). Let $Seq(E, \succ_\Gamma)$ denote the sequence of edges from $E$ according to the edge order $\succ_\Gamma$. Then for each edge $(v, u)$ in this sequence $Seq(E, \succ_\Gamma)$, if the degree of a node $v$ is greater than projection parametet $\theta$, I remove the edge $(v, u)$ from $E^\theta$ and also decrease the degree count of nodes $v$ and $u$ in *deg* list by 1 (Lines 2-4). Finally, the algorithm returns a $\theta$-bounded graph $G^\theta$ (Line 5).

**Example 7.** *Assume that a two-level ordering $\Gamma = (\succ_N, \succ_V)$ on a graph $G$ in Figure 7.3 is obtained by sorting nodes based on degrees from highest to lowest ($\succ_V$), and for each node $v$ sorting their neighbours in $N_G(v)$ in a similar manner ($\succ_N$). Thus, I have a sequence of edges ordered by $\succ_\Gamma$, i.e., $\langle (C, A), (C, D), (C, E), (C, F), \ldots, (F, C) \rangle$. Then, following this sequence, by checking whether $deg(C) > \theta$,* SER *first removes edge $(C, A)$ and decreases the degree counts of nodes $C$ and $A$ by 1. Similarly,* SER *removes edges $(C, D)$ and then the other edges if their node degree counts is greater than $\theta$ until $G^\theta$ is obtained.*

*SER* generates $\theta$-bounded graphs that are maximal in the sense that no edge from $E - E^\theta$ can be added into such $\theta$-bounded graphs without making their maximal node degree be above $\theta$. However, *SER* does not guarantee that these $\theta$-bounded graphs are optimal, i.e., keeping the largest possible number of edges, because *SER* depends on the ordering $f$ which may be locally optimal.

**Algorithm 6:** $\theta$-$dK\varepsilon$ algorithm

```
Input: A graph $G = (V, E)$;
       a privacy parameter $\varepsilon$
Output: A perturbed $\widehat{dK}$
1 $G^\theta \leftarrow$ Project $G$ by Algorithm 5
2 $dK^\theta \leftarrow$ Query $G^\theta$ with $f^{2K}$
3 $\widehat{dK} \leftarrow$ Perturb $dK^\theta$ w.r.t. Def. 26
4 Return $\widehat{dK}$
```

### 7.4.2 Releasing $dK$-distribution via Projection

Given a graph $G$, as shown in $\theta$-$dK\varepsilon$ Algorithm, instead of extracting a $dK$-distribution from an original graph $G$ directly, I extract a $dK$-distribution from a $\theta$-bounded graph $G^\theta$ (Line 1) which is transformed from $G$ by a graph projection algorithm $\mathcal{P}$ (Line 2). In this work, $\mathcal{P}$ refers to the *SER* algorithm. By Lemma 10 and the fact that the maximum degree in $G^\theta$ is no larger than $\theta$, I have the following lemma about the sensitivity of $f^{dK} \circ \mathcal{P}$.

**Lemma 8.** *Let $G \overset{n}{\sim} G'$ be two node neighboring graphs. Then the sensitivity of $f^{dK} \circ \mathcal{P}$ is upper bounded by $(2\theta + 1) \times \theta$, where $d = 2$.*

Based on the sensitivity of $f^{dK} \circ \mathcal{P}$, the perturbation is performed over the $dK$-distribution being extracted from $G^\theta$ to generate a $\varepsilon$-differentially private joint degree distribution (Line 3). A high-level description of this algorithm of releasing differentially private joint degree distribution via projection, namely $\theta$-$dK\varepsilon$, is presented in Algorithm 6. Since the perturbation in Algorithm 6 is conducted using the Laplace mechanism based on the sensitivity of $f^{dK} \circ \mathcal{P}$, I have the following lemma regarding the privacy guarantee of $\theta$-$dK\varepsilon$.

**Lemma 9.** *$\theta$-$dK\varepsilon$ generates $\varepsilon$-node-differential private dK-distribution.*

## 7.5 Experiments

In this section, I have conducted experiments to evaluate the proposed approach and discussed the experimental results.

### 7.5.1 Experimental Setup

**Datasets.** I used four network datasets in the experiments from different domains including social network and citation networks:

1. *Facebook*[1] is a network of social interactions and personal relationships, containing 4,039 nodes and 88,234 edges.

[1]Network datasets are available at http://snap.stanford.edu/data/index.html

**Table 7.1:** Comparison on the preserved edge ratio $|E^\theta|/|E|$ of $EAD$ and the proposed $SER$ graph projection approach under different values of $\theta$.

| Dataset | $\theta = 16$ | | $\theta = 32$ | | $\theta = 64$ | | $\theta = 128$ | | $\theta = 256$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EAD | SER | EAD | SER | EAD | SER | EAD | SER | EAD | SER |
| *Facebook* | 0.27 | 0.61 | 0.44 | 0.71 | 0.66 | 0.84 | 0.88 | 0.96 | 0.97 | 0.98 |
| *Wiki-Vote* | 0.19 | 0.59 | 0.32 | 0.66 | 0.50 | 0.76 | 0.71 | 0.87 | 0.88 | 0.96 |
| *Ca-HepPh* | 0.16 | 0.61 | 0.24 | 0.68 | 0.31 | 0.77 | 0.39 | 0.84 | 0.46 | 0.96 |
| *Email-Enron* | 0.17 | 0.52 | 0.22 | 0.61 | 0.29 | 0.71 | 0.36 | 0.80 | 0.43 | 0.89 |

2. *Wiki-Vote*[1] is a voting network of Wikipedia users, containing 7,115 nodes and 103,689 edges.
3. *Ca-HepPh*[1] is a scientific collaborative networks between authors and papers, containing 12,008 nodes and 118,521 edges.
4. *Email-Enron*[1] is a Email communication network from Enron, containing 36,692 nodes and 183,831 edges.

**Baseline methods.** I first compared the projection method, *Stable-Edge-Removal (SER)*, with the state-of-the art graph project method *Edge-Addition (EAD)* [14]. Then, I compared the utility of the following methods for generating differentially private $dK$-distributions:

1. *SER-$\theta$-dK$\varepsilon$*, which applies the proposed *$\theta$-dK$\varepsilon$* algorithm on projected graphs generated by SER.
2. *EAD-$\theta$-dK$\varepsilon$*, which applies the proposed *$\theta$-dK$\varepsilon$* algorithm on projected graphs generated by EAD.
3. $\varepsilon$-DP, which is a standard $\varepsilon$-differential privacy algorithm in which noise is added on an original graph using the Laplace mechanism [29].

**Utility metrics.** Following [14; 53; 84], I use three utility metrics:

1. *Preserved edge ratio* $|E^\theta|/|E|$ measures the ratio of edges being preserved by graph projection, where $|E|$ and $|E^\theta|$ denote the number of edges before and after applying graph projection, respectively.
2. *L1 distance* (or *L1 error*) measures the network structural error between an original $dK$-distribution $p$ and its perturbed $dK$-distribution $p'$, i.e.,

$$\|p - p'\|_1 = \sum_{j=1}^{deg(G)} \sum_{i=1}^{deg(G)} |p(i,j) - p'(i,j)| \tag{7.2}$$

where I pad entries for degree pairs $(i, j)$ with 0 if such degree pairs do not exist in $p$ or $p'$

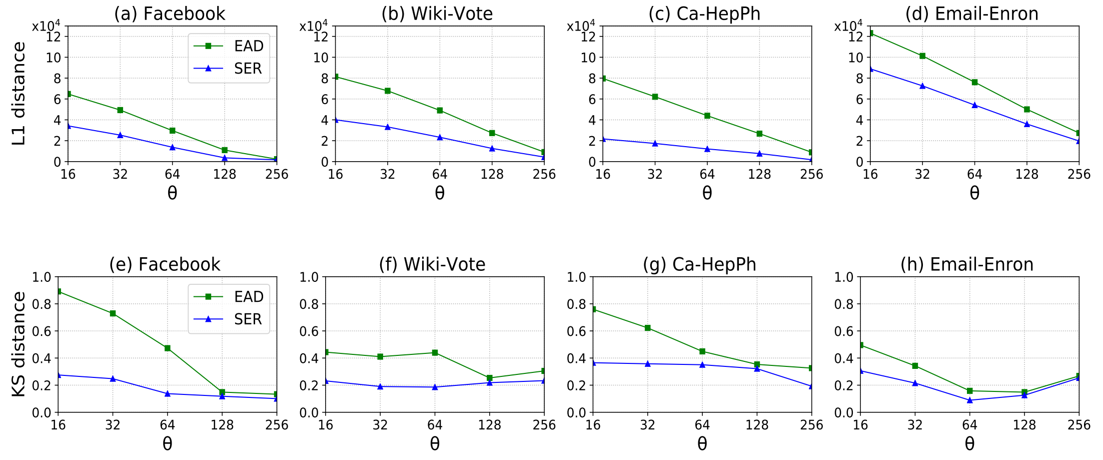


**Figure 7.4:** Comparison of graph project methods under varying $\theta$ over four datasets: (a)-(d) L1 distance and (e)-(h) KS distance.

3. *Kolmogorov-Smirnov distance* (or *KS distance*) quantifies the closeness between an original $dK$-distribution $p$ and its perturbed $dK$-distribution $p'$, i.e.,

$$KS(p, p') = max_i |CDF_{p(i,j)} - CDF_{p'(i,j)}| \tag{7.3}$$

where $CDF_{p(i,j)}$ is the value of cumulative distribution function on degree pairs $(i, j)$ from distribution $p$.

**Parameter settings and others.** For the privacy parameter $\varepsilon$, I choose $\varepsilon \in [0.01, 10.0]$ which cover the range of differential privacy levels widely used in the literature [46; 47]. For the projection parameter $\theta$, I followed [84; 14; 16] to choose $\theta \in \{1, 2, 4, \ldots, 2^{2log_2(|V|)}\}$. I used *Orbis* [71] to generate 2K-distributions.

### 7.5.2 Experimental Results and Discussion

**Evaluating graph projection.** I first compared the method *SER* with the state-of-the-art graph projection method *EAD* [14].

*(1) Preserved edge ratio.* Table 7.1 shows the results for preserved edge ratio. For every value of $\theta$, *SER* outperforms *EAD* by preserving more edges over all four datasets. This is consistent with the discussion in Section that the two-level ordering can generally preserve more edges than a random edge ordering.

*(2) L1 distance.* Figure 7.4(a)-(d) presents the results for L1 distance. For all four datasets, the projection method *SER* leads to less network structural error for every value of $\theta$ as compared to *EAD*. This verifies that *SER* can better preserve topological structure of a graph than *EAD* and maintain the shape of distribution after projection. Thus, the utility of projected $dK$-distribution by *SER* is higher as compared to *EAD*.

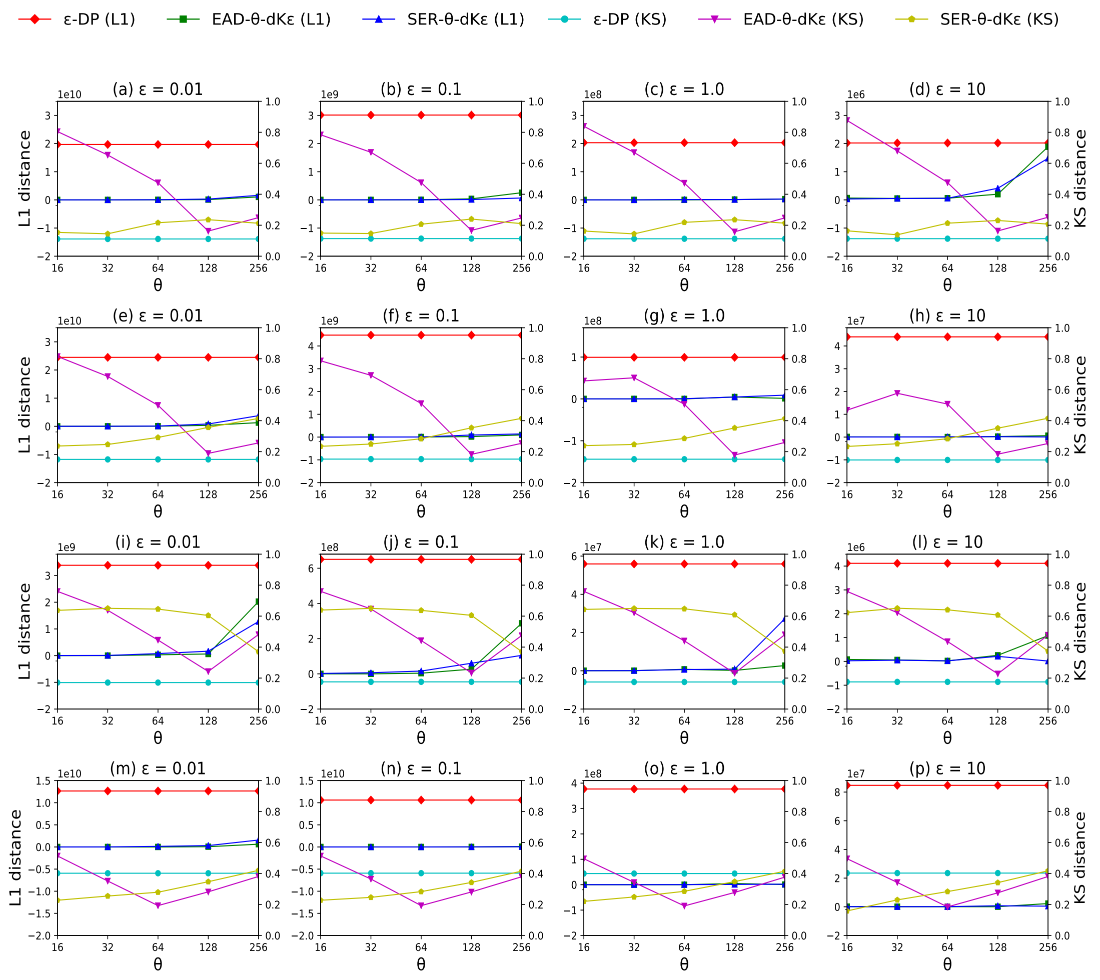


**Figure 7.5:** Comparison on the L1 distance (left) and KS distance (right) of differentially private $dK$-distributions under varying $\theta$, and $\varepsilon$ over four datasets: (a)-(d) *Facebook* dataset, (e)-(h) *Wiki-Vote* dataset, and (i)-(l) Ca-HepPh dataset, and (m)-(p) *Email-Enron* dataset.

*(3) KS distance.* Figure 7.4(e)-(h) presents the results for KS distance. I can see that, for every value of $\theta$, the projection method *SER* outperforms *EAD* with smaller KS-distances over all four datasets. Thus, projected $dK$-distributions generated by *SER* are more similar to their original $dK$-distributions.

**Evaluating differentially private $dK$-distributions.** I compared the overall utility of differentially private $dK$-distributions generated by the method *SER-θ-dKε* against the baseline methods. Figure 7.5 presents the results.

*(1) L1 distance.* For all four datasets, the method *SER-θ-dKε* yields less network structural error than $\varepsilon$-DP for every value of $\varepsilon$ and $\theta$. Compared to $\varepsilon$-DP, the results of *SER-θ-dKε* and *EAD-θ-dKε* are close and both much less. This is because, by approximating $f^{dK}$ to $f^{dK} \circ \mathcal{P}$ via a graph projection $\mathcal{P}$, two kinds of errors are introduced: one is random noise to guarantee node-DP and the other one is due to projection. I

notice that, the first kind of error, which depends on the sensitivity of $f^{dK} \circ \mathcal{P}$, dominates the impact on the utility of differentially private $dK$-distributions generated w.r.t. L1 distance. Thus, by reducing the sensitivity of a $dK$ function via projection, the amount of random noise being added to achieve node-DP is reduced significantly and the L1 errors of differentially private $dK$-distributions are thus also reduced significantly.

*(2) KS distance.* I observe that $\varepsilon$-DP outperforms *SER-θ-dKε* and *EAD-θ-dKε* for smaller datasets, except for the largest network *Email-Enron*. This is because, a projection method may change the topological structure of an original graph. However, for large networks such as *Email-Enron*, due to their high sensitivity, *SER-θ-dKε* and *EAD-θ-dKε* generally perform better than $\varepsilon$-DP. In addition, for smaller values of $\theta$, differentially private $dK$-distributions generated by *SER-θ-dKε* are more similar to their original $dK$-distributions than *EAD-θ-dKε*. However, for larger $\theta$ values, the results of *SER-θ-dKε* and *EAD-θ-dKε* are close. This is because for larger $\theta$ values the amount of noise injected to achieve DP is high, and the fact that *SER-θ-dKε* preserves more edges leads to more noise being added to frequency values of degree pairs in dK-distributions.

## 7.6 Summary

In this chapter, I have studied the problem of publishing higher-order network statistics such as *joint degree distribution* under node-DP. I have theoretically analyzed the sensitivity for publishing joint degree distribution and proposed a novel projection-based method in order to enhance the utility of released network statistics under node-DP. The effectiveness of the work has been empirically verified over four real-world networks.

Chapter 8

# dK-Personalization: Publishing Network Statistics with Personalized Differential Privacy

## 8.1 Overview

In this chapter, I study the problem of publishing network data distributions, particularly *degree distribution*, and *joint degree distribution*, which guarantees personalized (edge or node) differential privacy while enhancing network data utility. To the best of my knowledge, there is no work which considers both variations of differential privacy while considering personalization. I incorporate personalization into differential privacy mechanism such that individuals (nodes) in a network can specify their own privacy preference (i.e., degree of privacy protection) to guarantee personalized differential privacy (PDP) under edge-DP and node-DP. Undertaking the problem of releasing network data distributions under PDP brings up two challenges:

(i) Each node in a network has its own privacy preference in personalized settings whereas each data point in data distribution reflects information about more than one node.

(ii) Network data is highly sensitive to structural changes under DP. To address these challenges, I propose *four* PDP mechanisms and introduce *degree queries* for controlled sensitivity.

To address these challenges, I propose *four* PDP mechanisms and present *degree queries* for controlled sensitivity. To the best of my knowledge, this is the first study to publish network data distributions under PDP, while enhancing network data utility.

The main contributions of this chapter are as follows:

- I show how to publish network data distributions in a personalized differentially private manner under edge-DP and node-DP.
- I analyse the sensitivity of *degree queries* for publishing *degree distribution*, and *joint degree distribution* under edge-DP and node-DP

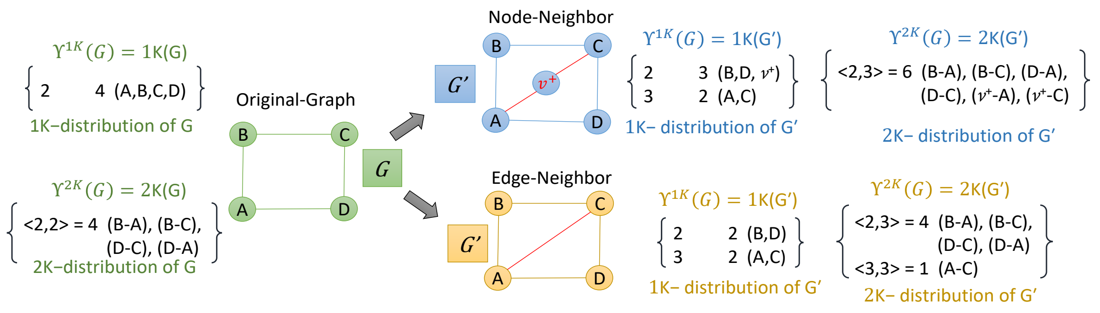

$\Upsilon^{1K}(G)$ = 1K(G)
2 4 (A,B,C,D)
1K–distribution of G
$\Upsilon^{2K}(G)$ = 2K(G)
<2,2> = 4 (B-A), (B-C), (D-C), (D-A)
2K–distribution of G
Original-Graph
G
Node-Neighbor
G′
$\Upsilon^{1K}(G)$ = 1K(G′)
2 3 (B,D, $v^+$)
3 2 (A,C)
1K– distribution of G′
$\Upsilon^{2K}(G)$ = 2K(G′)
<2,3> = 6 (B-A), (B-C), (D-A), (D-C), ($v^+$-A), ($v^+$-C)
2K– distribution of G′
Edge-Neighbor
G′
$\Upsilon^{1K}(G)$ = 1K(G′)
2 2 (B,D)
3 2 (A,C)
1K– distribution of G′
$\Upsilon^{2K}(G)$ = 2K(G′)
<2,3> = 4 (B-A), (B-C), (D-C), (D-A)
<3,3> = 1 (A-C)
2K– distribution of G′

**Figure 8.1:** An illustrative example of two (edge or node) neighboring graphs $G \sim G'$ with their corresponding *dK-distributions*, for $d = 1$, and $d = 2$.

- I introduce four approaches for generating personalized differentially private network data distributions while enhancing utility.
- I conduct comprehensive experiments over four real-world networks, and the results demonstrate that the proposed approaches can effectively enhance the utility of differentially private network data distributions with personalization.

Informally, personalization refers to different privacy levels chosen by different individuals based on their own preferences. For instance, in social networks an individual (user) tends to share their personal information with their close ones and only share obscured data with acquaintances or strangers. Thus, having different privacy levels based on their preferences. Personalized differential privacy (PDP) [29] have been proposed [49; 2; 32] to provide freedom to each individual to have a personalised privacy preference.

## 8.2 Problem Definition

Personalization allows each $v \in V$ to specify its own *privacy preference* $\varepsilon$ in $G$. However, given two (edge or node) neighboring graphs $G \sim G'$ where $G'$ is obtained from $G$ by adding (or deleting) an edge (or node) can affect more than one node. Thus, PDP should be formalized in terms of all affected nodes to guarantee $\varepsilon$-indistinguishability. In a privacy specification $\Phi = \{\varepsilon_1, \dots \varepsilon_n\}$, denote $\Phi^v$ the privacy preference $\varepsilon$ of a node $v$. For edge neighboring graphs $G \overset{e}{\sim} G'$, adding (or deleting) an edge affects exactly two nodes $u$ and $v$; likewise, for node neighboring graph $G \overset{n}{\sim} G'$, adding (or deleting) a node $v^+$ affects $|E^+|$ nodes incident to $v^+$ and $v^+$ itself. I formally define edge and node PDP below.

**Definition 30.** (Edge $\Phi$-PDP) *A randomized mechanism $\mathcal{K}$ satisfies edge $\Phi$-personalized differential privacy, if for each pair of edge neighboring graphs $G \overset{e}{\sim} G'$, and all possible outputs $\mathcal{O} \subseteq range(\mathcal{K})$, it holds:*

$$Pr[\mathcal{K}(G) \in \mathcal{O}] \leq e^{min\{\Phi^u,\Phi^v\}} \times Pr[\mathcal{K}(G') \in \mathcal{O}].$$

**Definition 31.** (Node $\Phi$-PDP) *A randomized mechanism $\mathcal{K}$ satisfies node $\Phi$-personalized differential privacy, if for each pair of node neighboring graphs $G \overset{n}{\sim} G'$, and all possible outputs $\mathcal{O} \subseteq range(\mathcal{K})$, it holds:*

$$Pr[\mathcal{K}(G) \in \mathcal{O}] \leq e^{min\{\Phi^{v} | (v^{+},v) \in E^{+}\}} \times Pr[\mathcal{K}(G') \in \mathcal{O}] \quad and$$
$$Pr[\mathcal{K}(G) \in \mathcal{O}] \leq e^{\Phi^{v^{+}}} \times Pr[\mathcal{K}(G') \in \mathcal{O}]$$

Given a graph $G$, I represent its topology properties as $dK$-distributions [72], where $f^{dK}$ queries $dK$-distribution $s.t$ $f^{dK}(G) = dK(G)$, as describe in Chapter 6. Figure 8.1 provides an illustrative example of two (edge or node) neighboring graphs with their corresponding $dK$-distribution for $d = 1$, and 2. In the work, for each $dK$-distribution $D$, I want to generate $D_{\Phi}$ that is an anonymized version of $D$ satisfying (edge or node) $\Phi$-PDP. Thus, I view the response to $f^{dK}$ for $d = 1$ and 2 as a collection of responses to degree queries, one for each tuple (entry) in a $dK$-distribution. Recall the definition of degree query presented in Chapter 6.

**Definition 32.** (Degree query) *A* degree query $f_q : f^{dK}(G) \rightarrow \mathbb{N}$ *maps a degree tuple $t \in f^{dK}(G)$ to a frequency value in $\mathbb{N}$ s.t. $(t, f_q(G)) \in f^{dK}(G)$.*

As DP implies PDP [49], to guarantee (edge or node) $\Phi$-PDP, I perform perturbation over real responses of $f_q$ by adding controlled noise from Laplace stochastic process [29]. The response to each $f_q$ can be combined into a complete $dK$-distribution using the parallel composition property of differential privacy [76]. More specifically, I add random noise into the real response $f_q(G)$, yielding a randomized response $f_q(G) + Lap(\Delta(f_q)/\varepsilon)$, where $\Delta(f_q)$ denotes the sensitivity of $f_q$ and $Lap(\Delta(f_q)/\varepsilon)$ denotes random noise drawn from a Laplace distribution. Here, $\varepsilon$ refers to a personalized privacy preference. I will discuss in detail how to utilize $\varepsilon$ to perform personalized perturbation in Section 8.4.

## 8.3 Sensitivity Analysis

The key challenge of releasing $dK$-distributions under PDP is to determine the right amount of noise that guarantees both privacy and accuracy. Unlike previous studies [85; 97; 47; 14; 45] that analyzed the entire $dK$-distribution sensitivity, I focus on the sensitivity of a single $dK$-distribution entry, i.e., *degree query $f_q$*.

**Sensitivity of $f_q$ for $1K$-distribution under node-DP.** Suppose that a node $v^{+}$ is added to $G$ with a set $E^{+}$ of new edges. Each edge in $E^{+}$ can cause at most one change in a current $1K$-distribution entry, so the total change in an entry is at most $|E^{+}|$. Furthermore, the new node $v^{+}$ can cause an additional change in the same entry. Thus, the maximum change in $f_q$ is at most $|E^{+}| + 1$.

**Lemma 10.** *Let $G \overset{n}{\sim} G'$ be two node neighboring graphs. When $d = 1$, $f_q(G)$ and $f_q(G')$ differ by at most $|E^{+}| + 1$.*

In the worst case, $|E^+|$ can be $|V|$. Thus, the total number of changes in $f_q(G)$, for $d = 1$, by adding a node is upper bounded by $|V| + 1$.

**Sensitivity of $f_q$ for $2K$-distribution under node-DP.** For a $2K$-distribution, when it is connected to a maximum degree node, each new edge $(v^+, v_i)$ in $E^+$ can affect at most $deg(G)$ edges. If all the $v_i$s incident to $v^+$ have the same degree, then the maximum increase in an entry is $deg(G) \times |E^+|$. This can be further increased if $v^+$ has the same degree as all the $v_i$s. Hence, the maximum change in $f_q$ is at most $(deg(G) + 1) \times |E^+|$.

**Lemma 11.** *Let $G \overset{n}{\sim} G'$ be two node neighboring graphs. When $d = 2$, $f_q(G)$ and $f_q(G')$ differ by at most $(deg(G) + 1) \times |E^+|$.*

Prior studies [59; 85] have shown that, in large social networks, $deg(G)$ is upper bounded by $O(\sqrt{|V|})$. Thus, for such networks, the sensitivity of $f_q$ for $d = 2$ of two node neighboring graphs is upper bounded by $O(|V|^{3/2})$.

**Sensitivity of $f_q$ for $1K$-distribution under edge-DP.** This is straightforward, because each new edge can affect at most two nodes.

**Lemma 12.** *Let $G \overset{e}{\sim} G'$ be two edge neighboring graphs. When $d = 1$, $f_q(G)$ and $f_q(G')$ differ by at most 2.*

**Sensitivity of $f_q$ for $2K$-distribution under edge-DP.** This is similar to the node-DP case. But since only one edge is added, it can affect at most $2 \times deg(G)$ existing edges. The new edge itself can further increase the entry by 1 when its end nodes have the same degrees as the those of the affected edges.

**Lemma 13.** *Let $G \overset{e}{\sim} G'$ be two edge neighboring graphs. When $d = 2$, $f_q(G)$ and $f_q(G')$ differ by at most $2 \times deg(G) + 1$.*

Prior studies [59; 85] have shown that, in large social networks, $deg(G)$ is upper bounded by $O(\sqrt{|V|})$. Thus, for such networks, the sensitivity of $f_q$ for $d = 2$ is upper bounded by $O(\sqrt{|V|})$ too.

I have observed that, by adding a single node or edge in $G$, the maximum change induced in the entries of $G$'s corresponding $dK$-distribution is two times greater than the maximum change induced in a single entry of a $dK$-distribution. Thus the sensitivity of *degree query* $f_q$, is half as compared to *dK-function* $f^{dK}$. This shows that the analysis as compared to analysis in existing studies [85; 97; 46; 14; 47] significantly enhances utility of published $dK$-distribution.

## 8.4 Proposed Personalized Approaches

In this section, I present *four* mechanisms for generating personalized differentially private $dK$-distribution for $d = 1$ and $d = 2$.

### 8.4.1 Local Least Based Personalized Perturbation

A straightforward perturbation can be done over $dK$-distribution $D$ by invoking the Laplace mechanism with the strongest privacy preference in $G$. This will, however, overprotect some individuals and degrade the output utility. To address this issue, I propose *Local Least (LL-dK)* mechanism to perturb the entries of $D$ with the strongest *local* privacy preference. More specifically, *LL-dK* perturbs each entry $x_i \in D$ with the smallest privacy preference $\varepsilon_{x_i}$ associated the corresponding nodes for $x_i$. For instance, in Figure 8.2 the frequency $p(1) = 2$ in $1K(G)$, and the frequency $p(2,4) = 3$ in $2K(G)$ are perturbed with the privacy preference $\varepsilon = min(\Phi^B, \Phi^F)$, and $\varepsilon = min(\Phi^A, \Phi^C, \Phi^D, \Phi^E)$, respectively.

In *LL-dK*, the perturbation is conducted using the Laplace mechanism based on the sensitivity of $f_q$, and personalized privacy preferences $\varepsilon_{x_i}$ Thus, by the parallel composition property of DP [76] I have the following lemma.

**Lemma 14.** *LL-dK generates $(\max_i \varepsilon_{x_i})$-personalized differentially private dK-distributions.*

As *LL-dK* uses the local strongest privacy preferences, which may lead to adding excessive noise when nodes with strong privacy preference have high centrality. To control the amount of random noise and improve personalization, another option is to simply discard high degree privacy conscious nodes.

### 8.4.2 Threshold Projection Based Personalized Perturbation

The *threshold projection (TP-dK)* approach (Algorithm 7) first transforms a graph $G$ to a $\theta$-bounded graph $G^\theta$ with $\theta < deg(G)$ [14; 16], removes all nodes in $G^\theta$ with $\Phi^v < \mu$ to get $G^{\theta,\mu}$, where $min(\Phi^v) < \mu < max(\Phi^v)$ is a global threshold, and then perturbs entries of $f^{dK}(G^{\theta,\mu})$. Since I have $deg(G) \leq \theta$, the sensitivity of $f_q$ is reduced; likewise, with threshold $\mu$ all nodes with high privacy concerns are removed which results in adding less noise to release $D$ of $G^{\theta,\mu}$.

**Algorithm 7:** *TP-dK algorithm*

```
Input: A graph $G = (V, E)$;
       a value $d$ in $1, 2$;
       a threshold $\mu$;
       a projection parameter $\theta$;
       a projection algorithm $\mathcal{P}$
Output: a perturbed $\widehat{dK}$
1 $G^\theta \leftarrow$ Project $G$ by $\mathcal{P}$ w.r.t. $\theta$
2 $G^{\theta,\mu} \leftarrow$ Truncate $G^\theta$ w.r.t. $\mu$
3 $dK^{\theta,\mu} \leftarrow$ Query $G^{\theta,\mu}$ with $f^{dK}$
4 $\widehat{dK} \leftarrow$ Perturb $dK^{\theta,\mu}$
5 Return $\widehat{dK}$
```

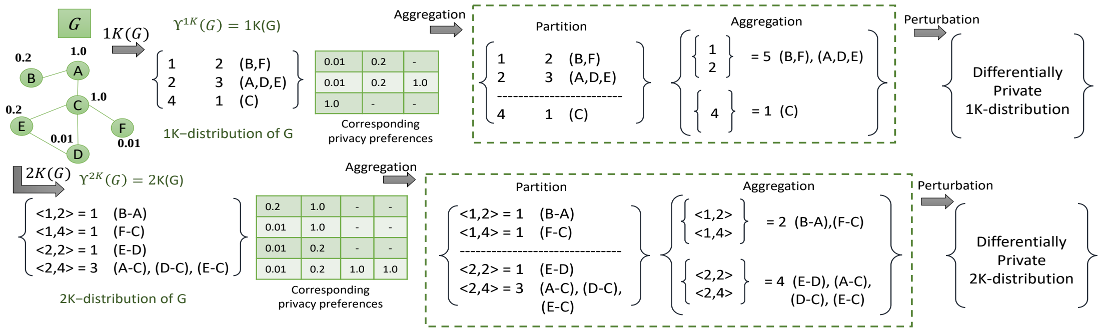


**Figure 8.2:** An illustrative example of aggregation based perturbation over *dK-distributions*, for $d = 1$, and $d = 2$.

In Algorithm 7, the perturbation is conducted using the Laplace mechanism (Line 4) based on the sensitivity of $f_q$, and personalized privacy preference $\mu$. Thus, by the parallel composition property of DP [76] I have the following lemma.

**Lemma 15.** *TP-dK generates $\mu$-personalized differentially private dK-distributions.*

The main challenge in this approach is to select good thresholds (i.e., $\theta$, and $\mu$) that can lower the sensitivity and preserve as much topological structure of a graph as possible [14]. To address this limitation, I now propose a personalized mechanism based on *sampling*.

## 8.4.3 Sampling Based Personalized Perturbation

Sampling is shown to be a powerful tool that can be integrated into differential privacy to amplify privacy protection [64; 49]. The *sampling (ST-dK)* approach (Algorithm 8) first splits each entry $x_i \in D$ such that frequency count of each $x_i$ is equal to one. Then samples each $x_i$ with probability $p_i = 1$ if $\varepsilon_{x_i} \geq \mu$, and samples other entries i.i.d. with probability $p_i = \frac{e^{\varepsilon_{x_i}}-1}{e^{\mu}-1}$ if $\varepsilon_{x_i} < \mu$, where $min(\Phi^v) < \mu < max(\Phi^v)$, and $\varepsilon_{x_i}$ is the smallest privacy preference associated the corresponding nodes for $x_i$. The inclusion probability for each $x_i$ depends on the corresponding $\varepsilon_{x_i}$, and the global threshold $\mu$.

**Algorithm 8:** *ST-dK algorithm*

```
Input: A graph $G = (V, E)$;
       a value $d$ in $1, 2$;
       a threshold $\mu$;
Output: a perturbed $\widehat{dK}$
1 $dK \leftarrow$ Query $G$ with $f^{dK}$
2 $dK^{\mu} \leftarrow$ Split and Sample $dK$
3 $\widehat{dK} \leftarrow$ Perturb $dK^{\mu}$
4 Return $\widehat{dK}$
```

In Algorithm 8, the perturbation is conducted using the Laplace mechanism (Line 3) over sampled $dK$-distribution based on the sensitivity of $f_q$, and personalized privacy preference $\mu$. Thus, by the parallel composition property of DP [76] I have the following lemma.

**Lemma 16.** *ST-dK generates μ-personalized differentially private dK-distributions.*

Our sampling mechanism is inspired by the results from [64; 49], where sampling prior to DP is shown to benefit privacy by combining two sources of randomness; however, introducing two kinds of errors: sampling error, and perturbation error. I observe that, smaller $\mu$ increases perturbation error, and larger $\mu$ increases sampling error, and vice versa. To reduce overall error, I now propose a personalized mechanism based on *aggregation*.

### 8.4.4 Aggregation Based Personalized Perturbation

The technique of aggregation is shown to reduce noise and enhance the utility of publishing differentially private histogram [14; 85]. Since degree distribution is obtained from normalizing the degree histogram [14], the idea of aggregation can be equivalently applied. the aggregation mechanism *AG-dK* (Algorithm 9) computes a table $\mathcal{T}$ by combining privacy preferences $\varepsilon$ of corresponding nodes associated with each $x_i \in D$ as a sorted list. Then, performing aggregation over $f^{dK}(G)$ consists of two steps: (i) entries with similar degrees and privacy preferences are partitioned into the same group; (ii) the frequency values of entries in the same group are aggregated. For instance in Figure8.2, a $dK$-distribution with $d = 1$, and $d = 2$ is partitioned into multiple groups, then the frequency values of entries in each group are aggregated by an aggregation process. The perturbation is performed over each aggregated frequency value with the smallest privacy preference $\varepsilon_{p_i}$ associated with the corresponding nodes for each partition $p_i$.

**Algorithm 9:** *AG-dK algorithm*

```
Input: A graph $G = (V, E)$;
       a value $d$ in $1, 2$;
       a partitioning parameter $k$;
       a partitioning algorithm $\mathcal{M}$
Output: a perturbed $\widehat{dK}$
1 $dK \leftarrow$ Query $G$ with $f^{dK}$
2 $\mathcal{T} \leftarrow$ Compute corresponding $\varepsilon$
3 $dK_i^p \leftarrow \mathcal{M}(dK)$ with $k$ and $\mathcal{T}$
4 $dK_i^a \leftarrow$ Aggregate $dK_i^p$
5 $\widehat{dK} \leftarrow$ Perturb $dK_i^a$
6 Return $\widehat{dK}$
```

In Algorithm 9, the perturbation is conducted using the Laplace mechanism (Line 5) over aggregated $dK$-distribution based on the sensitivity of $f_q$, and personalized privacy preferences $\varepsilon_{p_i}$ associated to each partition. Thus, by the parallel composition property of DP [76] I have the following lemma.

**Lemma 17.** *AG-dK generates* $(\max_i \varepsilon_{p_i})$*-personalized differentially private dK-distributions.*

*AG-dK* significantly reduces the total amount of noise to achieve PDP, particularly when the number of partitions in aggregated *dK*-distribution is smaller.

## 8.5 Experiments

In this section I have evaluated the proposed approaches, and discussed the results.

### 8.5.1 Experimental Setup

**Datasets.** I used four real-world network datasets from different domains including social, citation, and email networks:

1. *Facebook*[1] contains 4,039 nodes and 88,234 edges.
2. *Wiki-Vote*[1]contains 7,115 nodes and 103,689 edges.
3. *Ca-HepPh*[1] contains 12,008 nodes and 118,521 edges.
4. *Email-Enron*[1]contains 36,692 nodes and 183,831 edges.

**Privacy Specifications.** Following [49], I randomly divided nodes in a network into three groups: conservative with fraction $F_C = 0.54$, moderate with fraction $F_M = 0.37$, and liberal with fraction $F_L = 1.0 - (F_C + F_M)$, having high (randomly drawn from range $[\varepsilon_C, \varepsilon_M]$), medium (randomly drawn from range $[\varepsilon_M, \varepsilon_L]$), and low (fixed at $\varepsilon_L = 1.0$) privacy preferences, respectively. I used $\varepsilon_C, \varepsilon_M, \varepsilon_L \in [0.01, 1.0]$, which cover the range of DP levels used in the literature [46; 47; 49], where default values of $\varepsilon_C = 0.01$, and $\varepsilon_M = 0.2$ [49].

**Utility metrics.** Following [14; 16; 45], I used two utility metrics to measure the difference between the original *dK*-distribution $D$ and its private version $D_\Phi$:

1. *L1 distance* measures the network structural error by calculating

$$\|D - D_\Phi\|_1 = \sum_{i=1}^{deg(G)} |D_i - D_{\Phi_i}| \tag{8.1}$$

   where I pad a distribution entry with 0 if it does not exist in $D$ or $D_\Phi$

2. *Kolmogorov-Smirnov (KS) distance* measures the closeness between the cumulative distribution functions of $D$ and $D_\Phi$ by calculating

$$KS(D, D_\Phi) = max_i |CDF_{D_i} - CDF_{D_{\Phi_i}}| \tag{8.2}$$

[1]Network datasets are available at http://snap.stanford.edu/data/index.html

**Baseline methods.** I compared the utility of the following methods for generating personalized differentially private $dK$-distributions:

1. $PDP_{min}$ is a standard DP algorithm using Laplace mechanism [29] with the minimum privacy preference in a network [49].
2. *LL-dK* is the proposed *local least* approach.
3. *TP-dK* is the *threshold projection* approach which extends the projection algorithm Stable-Edge-Removal [45] for graph projection.
4. *ST-dK* is the *sampling* approach.
5. Additionally, I investigate two variations of *TP-dK*, and *ST-dK* with threshold [49] $\mu = max(\Phi^v)$, and $\mu = \frac{1}{n}\sum\Phi^v$ (i.e., the average privacy preference in a network). I denote these as *TP-dK*$_{avg}$, and *ST-dK*$_{avg}$.
6. *AG-dK* is the proposed *aggregation* based approach which extends the algorithm MDAV-*dK* [47] for partitioning $dK$-distributions.

**Parameter settings and others.** I choose $\theta \in \{1, 2, 4, \ldots, 2^{2log_2(|V|)}\}$ [14; 16], for projection in *TP-dK*. I vary sample size $m \in [30\%, 70\%]$ [65] to study the impact of sample size in *ST-dK*. Following [21; 47; 46], I choose $k \in \{2, 4, 6, \ldots, 100\}$ for partitioning in *AG-dK*. For each method, I ran 3 times and took the average result [46]. I use *Orbis* [71] to generate $dK$-distributions for $d = 1$ and $d = 2$.

### 8.5.2 Experimental Results and Discussion

**Evaluating personalized differentially private $1K$-distributions.** Figure 8.3 presents the experimental results. For all four datasets, w.r.t. L1 distance, under both edge-PDP and node-PDP, the methods yield less network structural error than $PDP_{min}$ for every value of $k$, $\theta$, and $m$, except for the largest network *Email-Enron*, where $PDP_{min}$ performs better than *ST-dK* and*ST-dK*$_{avg}$, when sample size is greater than 30%. This is because, by increasing the sample size for larger dataset, perturbation error dominates the impact on the utility. Overall, the methods *AG-dK*, *TP-dK*, and *TP-dK*$_{avg}$ outperform all other methods, because aggregation reduces the overall noise and projection reduces the sensitivity, thus enhancing output utility significantly. When measured by the KS distance, for all four datasets, under edge-PDP the method *AG-dK*, and under node-PDP the method *LL-dK* outperforms $PDP_{min}$ by generating more similar $1K$-distributions after perturbation. Overall, *TP-dK*, and *TP-dK*$_{avg}$ yield higher values of KS distance under both edge-PDP and node-PDP because graph transformation may change the topological structure of an original graph which results in generating less similar $dK$-distributions.

**Evaluating personalized differentially private $2K$-distributions.** Figure 8.4 presents the experimental results. For all four datasets, w.r.t. L1 distance, under both edge-PDP and node-PDP, the methods yield less network structural error than $PDP_{min}$ for

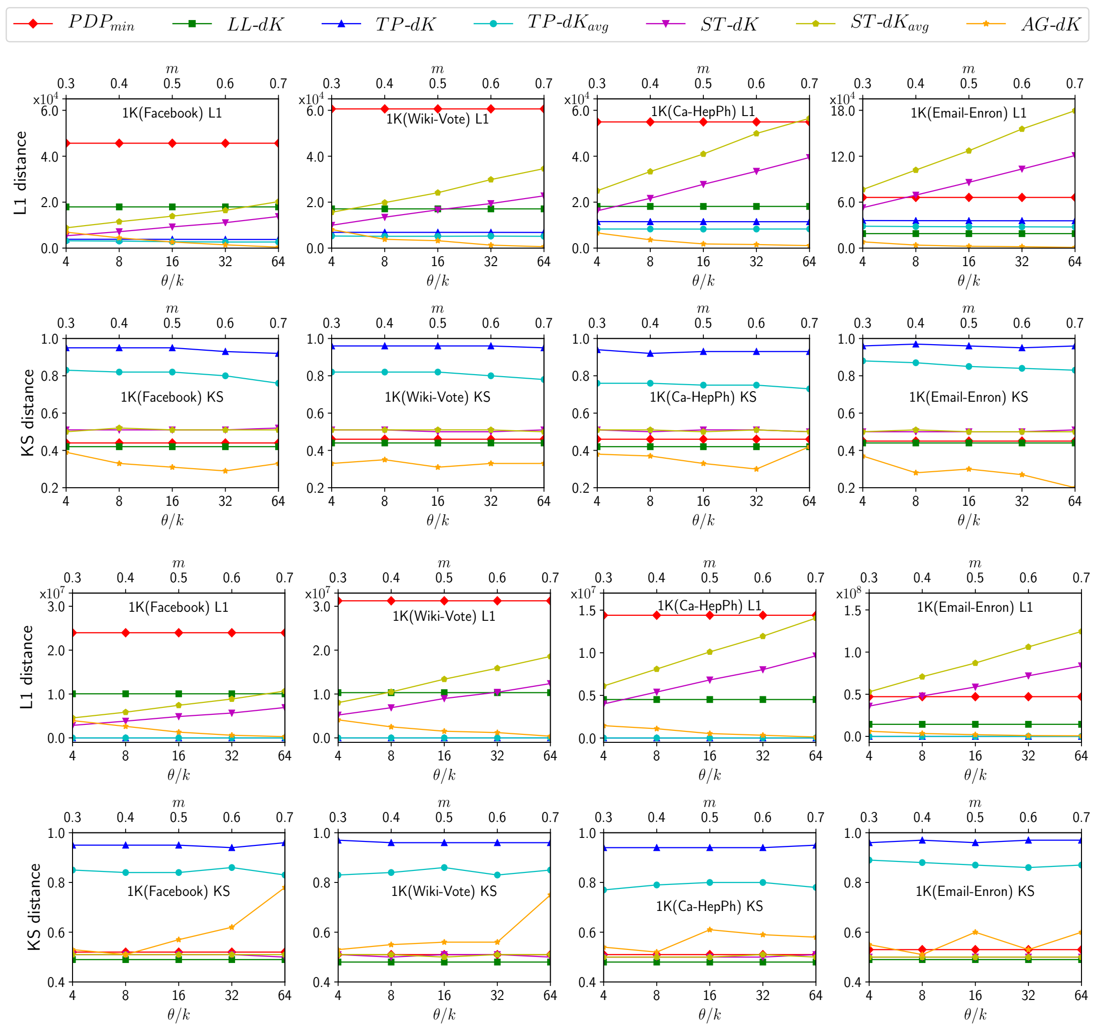


**Figure 8.3:** The L1 and KS distance of 1*K*-distributions on four datasets varying $\theta$, $k$, and $m$: (upper half) under edge-PDP and (lower half) under node-PDP.

every value of $k$, $\theta$, and $m$. Also, unlike to 1*K*-distribution, in 2*K*-distribution *ST-dK*, and *ST-dK*$_{avg}$ perform better than *LL-dK* because sensitivity of 2*K*-distributions is high as compared to 1*K*-distribution, thus overall more noise is added into 2*K*-distribution as compared to sampled 2*K*-distribution. Overall, the methods *TP-dK*, and *TP-dK*$_{avg}$ outperform all other methods, w.r.t. L1 distance, and the results of *AG-dK*, for larger $k$ values, are close to them, because aggregation and projection enhance the overall utility. On the other hand, w.r.t. KS distance, for all four datasets, under edge-PDP the method *AG-dK*, and under node-PDP the method *LL-dK* outperforms $PDP_{min}$ by generating more similar 2*K*-distributions after perturbation. In addition, under edge-PDP, the results of *LL-dK*, *ST-dK*, *ST-dK*$_{avg}$ and $PDP_{min}$ are close, which reflects both sampling error and perturbation error contribute towards the results. Contrary, under node-PDP, the results of *ST-dK*, *ST-dK*$_{avg}$ and $PDP_{min}$ are almost same which reflect the impact of high sensitivity, where perturbation error domi-

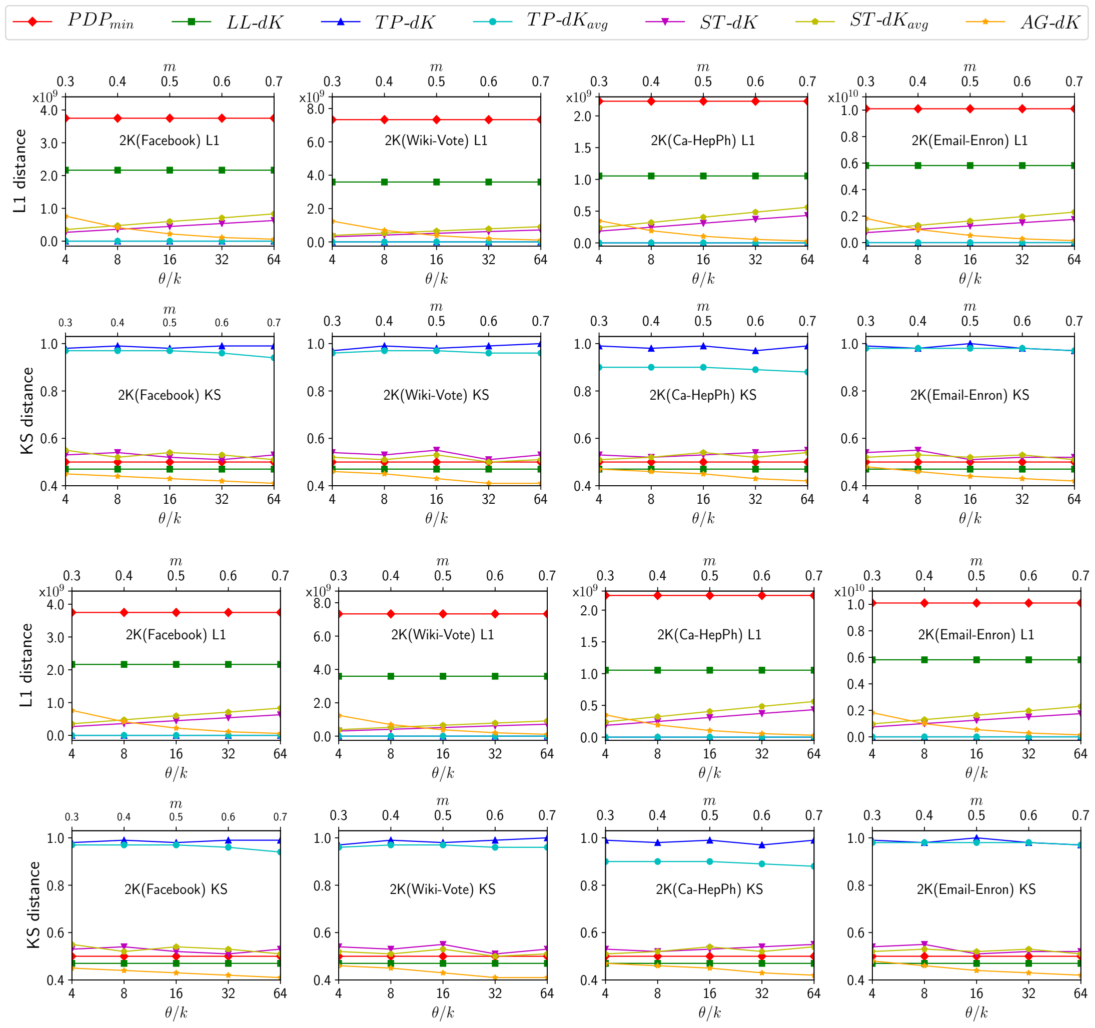


**Figure 8.4:** The L1 and KS distance of 2$K$-distributions on four datasets varying $\theta$, $k$, and $m$: (upper half) under edge-PDP and (lower half) under node-PDP.

nates. Overall, *TP-dK*, and *TP-dK*$_{avg}$ yield higher values of KS distance under both edge-PDP, and node-PDP because of graph transformation.

**Discussion.** I have analysed the trade-offs between utility and privacy of $dK$-distributions under PDP generated using the proposed approaches. I have noticed that, the error caused by random noise which depends on the sensitivity and the personalized privacy preference $\varepsilon$ dominates the impact on output utility. Increasing $\varepsilon$ and decreasing sensitivity can help to reduce error, though, generating more similar $dK$-distributions to their original $dK$-distribution is challenging due to noise addition. Reducing sensitivity can increase the output utility without compromising privacy; however, it is more challenging for node-PDP than for edge-PDP as graph data is highly sensitive under node-PDP.

## 8.6 Summary

In this chapter, I have studied the problem of publishing *degree distribution* and *joint degree distribution* under PDP. I have theoretically analyzed the sensitivity of these distributions under edge-PDP and node-PDP. I have proposed four personalized privacy-preserving mechanisms while enhancing output utility. The effectiveness of the proposed work has been empirically verified over four real-world networks.

Chapter 9

# Conclusions and Future Work

In this thesis, I have studied the problems that are related to privacy-preserving data publishing under the framework of differential privacy for different data structures, such as relational data and graph data, while significantly enhancing the utility of privacy-preserving data. First, I have explored privacy-preserving relational data publishing for which I have introduced a general new framework for publishing relational data under differential privacy, which is built upon microaggregation to enhance the data utility (Part 1: Chapter 4 - 5). I have further investigated privacy-preserving graph data publishing for which I have developed novel privacy-preserving data publishing mechanisms for publishing higher-order graph statistics under edge, node, and personalized differential privacy while enhancing output utility (Part 2: Chapter 6 - 8). In the following, I provided conclusions for two of these research areas.

## 9.1 Relational Data Publishing

In Chapter 4, I have studied the problem of generating $\varepsilon$-differentially private datasets by incorporating microaggregation into the relational data publishing process. I presented a microaggregation-based framework for generating $\varepsilon$-differentially private datasets and formulated the notion of *2-stable microaggregation* to characterize the correspondence of clusters in microaggregated datasets. I then proposed a stable microaggregation algorithm that can ensure the correspondence of clusters in the microaggregated datasets of two neighboring datasets. I finally verified the utility enhancement of the proposed framework on two real-world datasets containing numerical data. It shows that the algorithm can effectively enhance the utility of released data by providing better within-cluster homogeneity and reducing the amount of noise, in comparison with the state-of-the-art methods.

In Chapter 5, I have extended the preliminary work described in Chapter 4, and proposed a *general* framework called *$\alpha$-stable microaggregation* which generalized *stable microaggregation*. The general framework introduce a parameter (i.e., $\alpha$) to enforce particular correspondence among clusters in the microaggregated datasets of two neighboring datasets. In doing so, the amount of noise added to differentially private datasets can always be controlled by the parameter $\alpha$ and stable microaggregation becomes a special case of general framework with $\alpha = 2$. Conceptually, $\alpha$ indicates a trade-off between error introduced during microaggregation and error needed for achieving differential privacy. I further presented two algorithms under the proposed framework. I verified the utility and privacy trade-off in microaggregated and differentially private datasets achieved by the proposed framework on three real-world datasets containing numerical data. I showed in the experiments that the proposed framework outperforms the state-of-the-art methods in terms of preserving output utility while guarantee differential privacy.

For future work, one of the interesting directions to explore further is data-oriented microaggregation techniques [19] using locality-sensitive hashing (LSH) [83] to perform near-neighbor search [39] in order to find *similar* items (records) in a very large collection of items without looking at every pair to improve efficiency for larger datasets. The connection of LSH with the anonymization problem (i.e., the problem of grouping similar record and then anonymize each group in order to reduce information loss) can be used to address challenges of high dimensionality (two or more attributes) in the relational data setting while performing microaggregation. For instance, in relational data with high dimensions, nearest-neighbor search using LSH can be performed to form groups containing similar records by searching the nearest neighbor of sensitive records, and then aggregation and noise addition can be performed to achieve differential privacy. Hence, it would be interesting to extend and explore the applicability of the mechanisms proposed in Chapter 4 and Chapter 5 to data-oriented microaggregation in a way that the amount of noise needed to achieve differential privacy can be reduced if the noise is added to a microaggregated dataset, where microaggregation is done through a specially designed locality sensitive hashing (LSH) mechanism.

## 9.2 Graph Data Publishing

In Chapter 6, I have studied the problem of publishing anonymized graphs under guarantee of edge differential privacy. I proposed a microaggregation-based framework for graph anonymization which preserves the topological structures of an original graph at different levels of granularity while achieving edge differential privacy. I enhanced the utility of graph data by reducing the magnitude of noise needed to achieve differential privacy. Within the proposed framework, I further developed a simple yet effective microaggregation algorithm under a distance constraint. I have empirically verified the noise reduction and privacy guarantee of the proposed algorithm on three real-world graph datasets. I showed in the experiments that the proposed framework can significantly reduce noise added to achieve differential privacy over graph data, and thus enhance the utility of anonymized graphs.

In Chapter 7, I have studied the problem of publishing higher-order graph statistics under node differential privacy. I presented a novel framework to publish higher-order network statistics under node differential privacy. I have analysed the sensitivity of publishing joint degree distribution in the proposed framework, and introduced a new graph projection algorithm to reduce sensitivity of publishing network statistics under node differential privacy. I finally validated the effectiveness of the proposed algorithm over four real-world networks, and the results demonstrated that the proposed work can effectively enhance the utility of differentially private network statistics.

In Chapter 8, I have studied the problem of publishing network data distributions in a personalized differentially private manner under edge and node differential privacy. I have analysed the sensitivity of degree queries for publishing degree distribution, and joint degree distribution under edge and node differential privacy. Then, I introduced four approaches for generating personalized differentially private network data distributions while enhancing utility. I finally conducted experiments to validate the effectiveness of the proposed approaches over four real-world networks. The results demonstrated that the proposed approaches can effectively enhance the utility of differentially private network data distributions with personalization.

The research conducted on graph data can be used to explore the release of graph statistics in dynamic graphs. Thus, it would be interesting to explore: how well the graph statistics release methods proposed in Chapter 6 and Chapter 7 under edge and node differential privacy can perform for dynamic graphs. Moreover, topological structures of real-world networks (e.g., social networks) often remain unchanged but privacy preferences of individuals (nodes) are dynamically updated under dynamic conditions such as changing social status. Hence, it would be interesting to extend and explore the applicability of the personalized approaches proposed in Chapter 8 to dynamic graphs. Furthermore, the privacy-preserving graph data publishing mechanisms proposed in this thesis can be used to study in local settings, i.e., local differential privacy [52]. Another interesting direction to explore in future research is to release statistics about social groups in a graph while protecting privacy of individuals under zero knowledge privacy (ZKP) [37], as zero knowledge privacy is strictly stronger than the notion of differential privacy and is particularly attractive when modeling privacy in social networks.

# Bibliography

[1] Jemal H Abawajy, Mohd Izuan Hafez Ninggal, and Tutut Herawan. Privacy preserving social network data publication. *IEEE CST*, 18(3):1974–1997, 2016.

[2] Mohammad Alaggan, Sébastien Gambs, and Anne-Marie Kermarrec. Heterogeneous differential privacy. *JPC*, 7(2), 2016.

[3] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*, pages 181–190. ACM, 2007.

[4] C Benjamin, M Fung, K Wang, R Chen, and S Philip. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (Csur)*, pages 1–53, 2010.

[5] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *ITCS*, pages 87–96, 2013.

[6] Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Identity obfuscation in graphs through the information theoretic lens. *Information Sciences*, 275:232–256, 2014.

[7] L Burnett, K Barlow-Stewart, AL Proos, and H Aizenberg. The" genetrustee": a universal identification system that ensures privacy and confidentiality for human genetic databases. *Journal of Law and Medicine*, 10(4):506–513, 2003.

[8] Alina Campan and Traian Marius Truta. Data and structural k-anonymity in social networks. In *KDD*, pages 33–54. Springer, 2008.

[9] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. A survey of graph-modification techniques for privacy-preserving on networks. *Artificial Intelligence Review*, 47(3):341–366, 2017.

[10] James Cheng, Ada Wai-chee Fu, and Jia Liu. K-isomorphism: privacy preserving network publication against structural attacks. In *SIGMOD*, pages 459–470, 2010.

[11] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[12] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31. IEEE, 2012.

[13] Ramesh A Dandekar, Josep Domingo-Ferrer, and Francesc Sebé. Lhs-based hybrid microdata vs rank swapping and microaggregation for numeric microdata protection. In *Inference Control in Statistical Databases*, pages 153–162. Springer, 2002.

[14] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *SIGMOD*, pages 123–138, 2016.

[15] Xiaofeng Ding, Cui Wang, Kim-Kwang Raymond Choo, and Hai Jin. A novel privacy preserving framework for large scale graph data publishing. *TKDE*, 2019.

[16] Xiaofeng Ding, Xiaodong Zhang, Zhifeng Bao, and Hai Jin. Privacy-preserving triangle counting in large graphs. In *CIKM*, pages 1283–1292, 2018.

[17] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003.

[18] Josep Domingo-Ferrer, Antoni Martínez-Ballesté, Josep Maria Mateo-Sanz, and Francesc Sebé. Efficient multivariate data-oriented microaggregation. *The VLDB Journal*, 15(4):355–369, 2006.

[19] Josep Domingo-Ferrer and Josep Maria Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *TKDE*, 14(1):189–201, 2002.

[20] Josep Domingo-Ferrer and Jordi Soria-Comas. From t-closeness to differential privacy and vice versa in data anonymization. *Knowledge-Based Systems*, 74:151–158, 2015.

[21] Josep Domingo-Ferrer and Vicenç Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *KDD*, 11(2):195–212, 2005.

[22] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *ARES*, pages 990–993. IEEE, 2008.

[23] SN Dorogovtsev, JFF Mendes, et al. Evolution of networks: From biological nets to the internet and www. *OUP Catalogue*, 2013.

[24] George T Duncan and Diane Lambert. Disclosure-limited data dissemination. *JASA*, 81(393):10–18, 1986.

[25] Cynthia Dwork. Differential privacy. In *ICALP*, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

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

[27] Cynthia Dwork. Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340, 2011.

[28] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[29] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[30] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *FnT-TCS*, 9(3–4):211–407, 2014.

[31] Amalie Dyda, Michael Purcell, Stephanie Curtis, Emma Field, Priyanka Pillai, Kieran Ricardo, Haotian Weng, Jessica C Moore, Michael Hewett, Graham Williams, et al. Differential privacy for public health data: An innovative tool to optimize information sharing while protecting data confidentiality. *Patterns*, 2(12):100366, 2021.

[32] Hamid Ebadi, David Sands, and Gerardo Schneider. Differential privacy: Now it's getting personal. *POPL*, 50(1):69–81, 2015.

[33] Vladimir Estivill-Castro and Jianhua Yang. Fast and robust general purpose clustering algorithms. In *PRICAI*, pages 208–218, 2000.

[34] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *SIGMOD*, pages 211–222, 2003.

[35] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273. ACM, 2008.

[36] Tianchong Gao, Feng Li, Yu Chen, and XuKai Zou. Local differential privately anonymizing online social networks under hrg-based model. *TCSS*, 5(4):1009–1020, 2018.

[37] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards privacy for social networks: A zero-knowledge based definition of privacy. In *TCC*, pages 432–449, 2011.

[38] Minas Gjoka, Maciej Kurant, and Athina Markopoulou. 2.5 k-graphs: from sampling to generation. In *INFOCOM*, pages 1968–1976, 2013.

[39] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *ToC*, 8(1):321–350, 2012.

[40] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.

[41] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. In *PVLDB*, pages 102–114, 2008.

[42] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. *Computer science department faculty publication series*, page 180, 2007.

[43] Naoise Holohan, Spiros Antonatos, Stefano Braghin, and Pól Mac Aonghusa. $(k, \varepsilon)$-anonymity: $k$-anonymity with $\varepsilon$-differential privacy. *CoRR, abs/1710.01615*, 2017.

[44] Xueyang Hu, Mingxuan Yuan, Jianguo Yao, Yu Deng, Lei Chen, Qiang Yang, Haibing Guan, and Jia Zeng. Differential privacy in telco big data platform. *PVLDB*, 8(12):1692–1703, 2015.

[45] Masooma Iftikhar and Qing Wang. dk-projection: Publishing graph joint degree distribution with node differential privacy. In *PAKDD*, pages 358–370. Springer, 2021.

[46] Masooma Iftikhar, Qing Wang, and Yu Lin. Publishing differentially private datasets via stable microaggregation. In *EDBT*, pages 662–665, 2019.

[47] Masooma Iftikhar, Qing Wang, and Yu Lin. dk-microaggregation: Anonymizing graphs with differential privacy guarantees. In *PAKDD*, pages 191–203. Springer, 2020.

[48] Shouling Ji, Weiqing Li, Mudhakar Srivatsa, Jing Selena He, and Raheem Beyah. General graph data de-anonymization: From mobility traces to social networks. *TISSEC*, 18(4):12, 2016.

[49] Zach Jorgensen, Ting Yu, and Graham Cormode. Conservative or liberal? personalized differential privacy. In *ICDE*, pages 1023–1034. IEEE, 2015.

[50] Zach Jorgensen, Ting Yu, and Graham Cormode. Publishing attributed social graphs with formal privacy guarantees. In *SIGMOD*, pages 107–122, 2016.

[51] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *PVLDB*, 4(11):1146–1157, 2011.

[52] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SICOMP*, 40(3):793–826, 2011.

[53] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Analyzing graphs with node differential privacy. In *TCC*, pages 457–476. Springer, 2013.

[54] Daniel Kifer and Bing-Rong Lin. Towards an axiomatization of statistical privacy and utility. In *PODS*, pages 147–158. ACM, 2010.

[55] Jong Wook Kim, Beakcheol Jang, and Hoon Yoo. Privacy-preserving aggregation of personal health data streams. *PloS one*, 13(11):e0207639, 2018.

[56] Fragkiskos Koufogiannis, Shuo Han, and George J Pappas. Gradual release of sensitive data under differential privacy. *JPC*, 7(2), 2016.

[57] Fragkiskos Koufogiannis and George J Pappas. Diffusing private data over networks. *TCNS*, 5(3):1027–1037, 2017.

[58] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph. *KDD*, 11(3):243–271, 2005.

[59] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *WWW*, pages 591–600, 2010.

[60] Diane Lambert. Measures of disclosure risk and harm. *Journal of Official Statistics*, 9(2):313, 1993.

[61] Michael Laszlo and Sumitra Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005.

[62] Haoran Li, Li Xiong, Zhanglong Ji, and Xiaoqian Jiang. Partitioning-based mechanisms under personalized differential privacy. In *PAKDD*, pages 615–627. Springer, 2017.

[63] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, pages 106–115. IEEE, 2007.

[64] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *ASIACCS*, pages 32–33, 2012.

[65] Bing-Rong Lin, Ye Wang, and Shantanu Rane. On the benefits of sampling in privacy preserving statistical analysis on distributed databases. *CoRR, abs/1304.4613*, 2013.

[66] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *SIGMOD*, pages 93–106, 2008.

[67] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, pages 24–24. IEEE, 2006.

[68] Ashwin Machanavajjhala, Xi He, and Michael Hay. Differential privacy in the wild: A tutorial on current practices & open challenges. In *SIGMOD*, pages 1727–1730, 2017.

[69] Ashwin Machanavajjhala and Daniel Kifer. Designing statistical privacy for your data. *Communications of the ACM*, 58(3):58–67, 2015.

[70] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.

[71] Priya Mahadevan, Calvin Hubble, Dmitri Krioukov, Bradley Huffaker, and Amin Vahdat. Orbis: rescaling degree correlations to generate annotated internet topologies. In *SIGCOMM*, pages 325–336, 2007.

[72] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. Systematic topology analysis and generation using degree correlations. In *SIGCOMM*, pages 135–146, 2006.

[73] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, KC Claffy, and Amin Vahdat. The internet as-level topology: three data sources and one definitive metric. *SIGCOMM*, 36(1):17–26, 2006.

[74] Josep Maria Mateo-Sanz, Francesc Sebé, and Josep Domingo-Ferrer. Outlier protection in continuous microdata masking. In *Workshop on Privacy in Statistical Databases*, pages 201–215, 2004.

[75] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[76] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30. ACM, 2009.

[77] Yvonne Mülle, Chris Clifton, and Klemens Böhm. Privacy-integrated graph clustering through differential privacy. In *EDBT/ICDT Workshops*, pages 247–254, 2015.

[78] Shirin Nilizadeh, Apu Kapadia, and Yong-Yeol Ahn. Community-enhanced de-anonymization of online social networks. In *SIGSAC CCS*, pages 537–548. ACM, 2014.

[79] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *TOC*, pages 75–84, 2007.

[80] Anna Oganian and Josep Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the IAOS*, 18(4):345–353, 2001.

[81] Ismet Ozalp, Mehmet Emre Gursoy, Mehmet Ercan Nergiz, and Yucel Saygin. Privacy-preserving publishing of hierarchical data. *TOPS*, 19(3):7, 2016.

[82] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proceedings of the VLDB Endowment*, 7(8):637–648, 2014.

[83] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[84] Sofya Raskhodnikova and Adam Smith. Efficient lipschitz extensions for high-dimensional graph statistics and node private degree distributions. *CoRR, abs/1504.07912*, 2015.

[85] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Sharing graphs using differentially private graph models. In *SIGCOMM*, pages 81–98, 2011.

[86] David Sánchez, Josep Domingo-Ferrer, and Sergio Martínez. Improving the utility of differential privacy via univariate microaggregation. In *PSD*, pages 130–142. Springer, 2014.

[87] David Sánchez, Josep Domingo-Ferrer, Sergio Martínez, and Jordi Soria-Comas. Utility-preserving differentially private data releases via individual ranking microaggregation. *Information Fusion*, 30:1–14, 2016.

[88] Entong Shen and Ting Yu. Mining frequent graph patterns with differential privacy. In *SIGKDD*, pages 545–553, 2013.

[89] Jordi Soria-Comas and Josep Domingo-Ferrer. Differentially private data publishing via optimal univariate microaggregation and record perturbation. *Knowledge-Based Systems*, 153:78–90, 2018.

[90] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. Improving the utility of differentially private data releases via k-anonymity. In *TrustCom*, pages 372–379, 2013.

[91] Jordi Soria-Comas, Josep Domingo-Ferrer, David Sánchez, and Sergio Martínez. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. *The VLDB Journal*, 23(5):771–794, 2014.

[92] Latanya Sweeney. k-anonymity: A model for protecting privacy. *IJUFKS*, 10(05):557–570, 2002.

[93] Latanya Sweeney, José Antonio Alonso, and M Teresa Lamata. Achieving k-anonymity privacy protection using generalization and suppression. *IJUFKS*, 10(05):557–570, 2002.

[94] Yufei Tao, Xiaokui Xiao, Jiexing Li, and Donghui Zhang. On anti-corruption privacy preserving publication. In *ICDE*, pages 725–734. IEEE, 2008.

[95] Jonathan Ullman and Adam Sealfon. Efficiently estimating erdos-renyi graphs with node differential privacy. In *NeurIPS*, pages 3765–3775, 2019.

[96] K Wang, R Chen, BC Fung, and PS Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Computing Surveys (Csur)*, 2010.

[97] Yue Wang and Xintao Wu. Preserving differential privacy in degree-correlation based graph generation. *TDP*, 6(2):127–145, 2013.

[98] Yue Wang, Xintao Wu, and Leting Wu. Differential privacy preserving spectral graph analysis. In *PAKDD*, pages 329–340, 2013.

[99] Yue Wang, Xintao Wu, Jun Zhu, and Yang Xiang. On learning cluster coefficient of private networks. *SNAM*, 3(4):925–938, 2013.

[100] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. ($\alpha$, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *KDD*, pages 754–759. ACM, 2006.

[101] Qian Xiao, Rui Chen, and Kian-Lee Tan. Differentially private network data release via structural inference. In *SIGKDD*, pages 911–920, 2014.

[102] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *SIGMOD*, pages 229–240. ACM, 2006.

[103] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *TKDE*, 23(8):1200–1214, 2011.

[104] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *SDM*, pages 150–168. Springer, 2010.

[105] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Ge Yu. Differentially private histogram publication. In *ICDE*, pages 32–43. IEEE, 2012.

[106] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. Differentially private histogram publication. *The VLDB Journal*, 22(6):797–822, 2013.

[107] Shengzhi Xu, Sen Su, Li Xiong, Xiang Cheng, and Ke Xiao. Differentially private frequent subgraph mining. In *ICDE*, pages 229–240. IEEE, 2016.

[108] Shen Yan, Shiran Pan, Yuhang Zhao, and Wen-Tao Zhu. Towards privacy-preserving data mining in online social networks: Distance-grained and item-grained differential privacy. In *ACISP*, pages 141–157. Springer, 2016.

[109] William E Yancey, William E Winkler, and Robert H Creecy. Disclosure risk assessment in perturbative microdata protection. In *ICSD*, pages 135–152. 2002.

[110] Xiaojun Ye, Yawei Zhang, and Ming Liu. A personalized (a, k)-anonymity model. In *WAIM*, pages 341–348. IEEE, 2008.

[111] Xiaowei Ying, Kai Pan, Xintao Wu, and Ling Guo. Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In *SNMA*, pages 1–10, 2009.

[112] Xiaowei Ying and Xintao Wu. Randomizing social networks: a spectrum preserving approach. In *SDM*, pages 739–750, 2008.

[113] Mingxuan Yuan, Lei Chen, and Philip S Yu. Personalized privacy protection in social networks. *PVLDB*, 4(2):141–150, 2010.

[114] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *KDD*, pages 153–171. Springer, 2007.

[115] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, pages 506–515, 2008.

[116] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *PVLDB*, 2(1):946–957, 2009.