UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

From the Institute of Theoretical Computer Science
of the Universität zu Lübeck
Director: Prof. Dr. math. Rüdiger Reischuk

k-anonymous Microaggregation

Dissertation
for the Fulfillment of
Requirements for the
Doctoral Degree
of the Universität zu Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by

Florian Thaeter
from Rendsburg

Lübeck 2021

First referee:	Prof. Dr. Rüdiger Reischuk
Second referee:	Prof. Dr. Esfandiar Mohammadi
Date of oral examination:	27.09.2021
Approved for printing. Lübeck,	29.09.2021

Abstract

The subject of k-anonymous microaggregation is the most prominent way to anonymize microdata. Introduced in the early nineties, it offers an attractive trade-off between privacy of data providers and utility for scientific research. In short, microaggregation uses k-member clustering to cluster individuals into homogeneous clusters and releases aggregate vectors, representing these clusters. Privacy of individuals is improved compared to simple methods like pseudonymization, because re-identification attacks are severely limited by the concept of *hiding in a group of k*. Increasing utility, which is usually measured in terms of cluster homogeneity, has been the most important research interest in the past. In this thesis we take a look at the broader picture: Besides improving microaggregation in the traditional way by presenting algorithms that cause less distortion, we also introduce more time-efficient algorithms and clarify open questions about the theoretical complexity of microaggregation. Aside from these main achievements, this thesis offers a coherent mathematical model of anonymization throughout every aspect of microaggregation research and sets it into perspective with other anonymization methods and utility goals.

Zusammenfassung

Die k-anonyme Microaggregation ist die am weitesten verbreitete Möglichkeit Mikrodaten zu anonymisieren. Sie wurde in den frühen neunziger Jahren vorgestellt und bietet ein ausgewogenes Verhältnis zwischen Anonymität der enthaltenen Personen und der Datennützlichkeit für wissenschaftliche Untersuchungen. Kurz zusammengefasst verwendet die Microaggregation ein k-member Clustering um Individuen in homogene Cluster aufzuteilen und veröffentlicht dann repräsentative Aggregationsvektoren für jedes dieser Cluster. Die Anonymität der Individuen ist dabei im Vergleich zu simpleren Methoden wie der Pseudonymisierung erhöht, da Reidentifikationsattacken durch das Konzept *hiding in a group of k* erheblich erschwert werden. Das Erhöhen der Datennützlichkeit, welche üblicherweise anhand der Clusterhomogenität gemessen wird, war bisher das Hauptziel wissenschaftlicher Forschung. In dieser Arbeit wollen wir jedoch einen breiter aufgestellten Ansatz verfolgen: Verbesserungen an der Microaggregation im traditionellen Sinne, durch das Gestalten von Algorithmen, welche geringere Störungen induzieren, sind dabei nur der Anfang. Weiter führen wir neue, zeiteffizientere Algorithmen ein und klären offene Fragen bezüglich der theoretischen Zeitkomplexität der Microaggregation. Darüber hinaus bietet die vorliegende Dissertation ein mathematisches Modell der Anonymisierung, welches in allen Aspekten der Forschung an Microaggregation Anwendung finden kann und den Vergleich zu alternativen Anonymisierungs- und Datennützlichkeitsdefinitionen ermöglicht.

Danksagungen

An erster Stelle möchte ich Rüdiger Reischuk danken, der meine Promotion als Co-Autor, Mentor und Doktorvater nicht nur mit seiner fachlichen Kompetenz inhaltlich, sondern auch durch das Angebot einer vollen Promotionsstelle finanziell ermöglicht hat. Weiter gilt mein Dank den Professoren und sonstigen Kollegen am TCS, ITS und IFIS, welche mir stets wertvolle Anreize und Feedback zu neuen Forschungsideen gegeben haben. Insbesondere möchte ich Maciej Liśkiewicz, Esfandiar Mohammadi, Thomas Eisenbarth, Oliver Witt und Tanya Braun für den wissenschaftlichen Austausch in diversen Forschungstreffen danken. Nicht vergessen möchte ich die Studierenden, welche ich bei der Anfertigung ihrer Abschlussarbeiten unterstützen durfte: Vielen Dank an Finn, Gudrun, Ivo, Mustafa und Yara. Es war immer eine Freude mit euch zu arbeiten. Ein ganz besonderer Dank geht an meine Familie: Jacqueline, Meike, Hasko und Johanna, danke für alles!

Contents

1

Introduction

Scientific research brought upon countless improvements to our society. By observing the world and inferring results, humanity has been able to create technology improving nearly every aspect of our lives, from the advancement of medical treatment to transportation, urban planning and the economy as a whole. This progress has been and is only possible due to the availability of data to conduct studies observing and describing the world. Without data, there would be uncertainty. Microaggregation is a modern approach to one of the most important problems regarding the collection and publication of data: privacy. By applying this effective anonymization strategy we are able to publish so-called *microdata* without the need of strong access restrictions harming the utility.

Microdata is the most basic form of data collected to be used for scientific studies. In contrast to aggregated data, microdata contains specific information for each record individually. In microdata individuals are described by tuples of attributes, typically visualized by tables, in which rows describe the individuals and columns describe the attributes. In the context of microdata, an individual might be a real or fictional person, a company, or any other clearly defined entity. Attributes on the other hand might be unique identifiers, numerical or categorical values like a persons height, weight or blood type. The purpose of microdata is to perform statistical analyses to obtain information about dependencies between attributes of individuals as well as relative frequencies of individuals with certain attribute value combinations. For example, in medical data there could or could not be any links between a specific disease and the age or gender of a patient. Another example would be the effects on a person's income depending on their place of residence which could be learned from census data.

A major advantage of microdata compared to other forms of data is its universality. Once collected, it can be used in different studies, investigating different issues. However, its universality is also a disadvantage: By nature, microdata contains sensitive personal information about individuals which comes with the obligation of privacy protection in compliance with ethics, company guidelines and national laws such as the GDPR of the European Union [26] or the PIPEDA of Canada [33]. While the exact definition of personal data and allowed methods to protect it differ from context to context, a basic rule is that data that can be used to identify an individual must not be published without some form of access restriction. As access restrictions prohibit the intended use of microdata, it must be assured that individuals cannot be identified. Not only are individuals identifiable from data containing unique identifiers, but also re-identifiable if enough other information can be gathered. This is called a *re-identification attack*. Such attacks can be thought of as stalking

attacks by curious neighbors. This metaphor fits because it is assumed that neighbors know their target quite well as they know a lot of attributes like (approximate) age, address, gender, number of kids, and so on. In a less metaphorical way, the stalking neighbor describes the combination of data from different sources, often referred to as *background knowledge attacks*. At this point, the main problem analyzed in this thesis arises: How can microdata be anonymized properly without losing the possibility of wide-range scientific evaluation?

The demand for availability of data as well as the possibility of large-scale data distribution have increased significantly over the last decades and so it has become more and more important to develop methods that allow the unrestricted publication of microdata, enabled through anonymization. In an ideal world, anonymized microdata allows the same scientific studies as the original microdata, but is not suitable to re-identify individuals or even infer any unfavorable information about small groups of people. Unfortunately, in practice, this perfect utility-preserving anonymization is impossible. Weak forms of anonymization like the replacement of unique identifiers by pseudonyms can easily lead to re-identification as demonstrated over and over by privacy scandals discovered by the media and privacy researchers. Three famous examples of insufficient anonymization are the the re-identification of Governor William Weld's medical information [6], the AOL incident [5] and the re-identification of anonymous movie reviews published by Netflix [56]. All these attacks use the fact that most people are unique given only a few harmless-looking attribute values, as demonstrated in [67] and [30]. However, reducing the data to coarse aggregates or publish study results instead, might preserve privacy but also defeats the purpose of microdata as a tool to share general purpose information.

Several techniques to find adequate trade-offs between anonymity and utility have emerged since the late-nineties of the 20th century. While most of them came and went unnoticed by the general public due to the sacrifice of too much utility and usability in the name of privacy, there are two competing frameworks that offer attractive trade-offs and slowly make their way into adaptation in more and more applications: *k-anonymous microaggregation*, which is designed for the anonymization of microdata as well as *differential privacy*, which is used mostly for the protection of interactive data queries.

Microaggregation which is short for microdata aggregation uses the mathematical concept of k-member clusterings. Small homogeneous clusters of individuals are created and for each cluster the size as well as single representative attribute vectors are published. To protect privacy, microaggregation relies on the concept of *hiding in a group of individuals*, which states that there is some form of re-identification protection when only aggregates of clusters with a certain minimum number of elements are reported. To protect utility, microaggregation clusters are small compared to the total number of individuals and are chosen data-dependent but independently of specific research goals.

Despite this specific use-case for k-member clustering, this thesis is not to be interpreted solely as a guideline for practical use of microaggregation. Instead, in the following chapters the problem will be analyzed mostly from the perspective of asymptotic complexity and comparative analysis of k-member clustering algorithms. As no efficient algorithms to find optimal solutions are known, average case complexity and real-world utility is evaluated by extensive experiments using benchmark databases. In the next chapter, a precise definition of microaggregation and all other relevant concepts is given. Further, notions

of anonymization cost are given and analyzed. Subsequently, chapter 3 sets microaggregation into perspective within the realm of other microdata privacy protection mechanisms. The challenges of non-numerical data are discussed alongside alternative utility units and stronger anonymity guarantees. Chapter 4 returns to a more theoretical approach to k-member clustering. Its main result is a complete proof of NP-hardness of microaggregation which was missing in literature before. Chapter 5, 6 and 7 focus on the development of improved microaggregation heuristics using three different sets of techniques to achieve equal amounts of privacy with less harm to utility. The algorithms `MDAV*` `ONA*` and `MONA`, which were developed during the work on this thesis and are presented in chapters 5 to 7, significantly improve upon previous state-of-the-art algorithms in the respective categories, which is demonstrated by extensive experimental analyses. The topic of heuristics is completed with chapter 8, which summarizes other microaggregation heuristics using unconventional techniques. The main part of this thesis concludes in chapters 9 and 10 which cover the topics of approximation guarantees and differential privacy in microaggregation, respectively. In chapter 9 it is analyzed, why there seems to be a lack of microaggregation algorithms for which worst case utility guarantees can be given, considering the large number of such algorithms for similar clustering problems like k-means clustering. Chapter 10 on the other hand gives a short introduction into differential privacy and discusses why there is no established replacement of microaggregation using this more popular anonymization method. Further, approaches to close this gap are discussed briefly. A short recap summarizing the achievements and unanswered questions of this thesis is given in chapter 11. Finally, the appendix consists of a list of publications extracted from the work on this thesis as well as a list of bachelor's and master's theses supervised during that time. Moreover, additional experimental results evaluating new microaggregation heuristics are given for completeness.

2

Anonymity through Clustering: A Mathematical Model

This chapter is dedicated to define microaggregation and to discuss some of its relevant properties used throughout the rest of this thesis. For this, we will take a look at databases, anonymization in general and microaggregation in particular.

2.1 Databases

Finding a one-fits-all definition for databases is a difficult task. As most clustering algorithms need some form of distance measure between elements, we would like to model databases as collections of elements drawn from a metric space. However, this might not cover all aspects of data. While distance measures for some categorical attributes like *biological gender* are straightforward, for other attributes like *profession* it might not be so easy. In this definition that is used used in most parts of the upcoming text we nevertheless assume d-dimensional numerical data. This choice allows us to apply a wide range of distance, anonymity and utility measures while being simple enough to keep definitional overhead low. In section 3.1 the possibility of non-numerical data is explored.

Another important detail is the form of accumulation of data. There is no consistent form to be found in the relevant literature. While sets are rarely used, we could use multisets like e.g. in [57, 61] or sequences like e.g. in [60, 39]. In the context of differential privacy a histogram-representation is commonly used (see [24]). In this main definition of data we assume databases to be sequences of data elements, as this is the most specific form of the aforementioned options. Whenever reasonable we might ignore the order of elements and thus consider multisets. Databases that do not allow duplicates will be modeled as sets when needed.

Definition 2.1 (Database).
A *database element* $x_i = (x_i^1, \ldots, x_i^d)$ is a d-dimensional tuple of numerical data drawn from $\mathbb{R}^d$. A *database* $X = x_1, \ldots, x_n$ is a sequence of database elements, potentially including duplicate elements. The set of all possible databases of n elements in d dimensions is denoted by $\mathcal{X}_{n \times d}$.

A small example for a database according to our definition is included as table 2.1.

Table 2.1: The numerical database *TestData* used throughout this text contains two (QI)-attributes and six elements. It is hence classified as a database $X \in \mathcal{X}_{6\times 2}$.

Element	Attribute 1	Attribute 2
x_1	3	6
x_2	6	3
x_3	9	3
x_4	9	3
x_5	12	6
x_6	12	9

We will often need to address certain parts of a database to describe algorithms, define anonymity and measure utility. The following notation is used for this purpose.

Definition 2.2 (Index Sets).
Concerning a database X, the *index set* $X_x := \{i | x_i = x\}$ contains all indices of elements with attribute tuple $x \in \mathbb{R}^d$ inside the sequence X.

In the context of k-anonymity it is assumed that data attributes come in three forms: *Identifiers*, *Quasi-Identifiers* and *Confidential Attributes*. For convenience these terms are abbreviated I, QI and CA throughout this text. The first step of any anonymization should be to remove all Identifiers. As these are unique identifying attributes like social insurance numbers or full names, any occurrence of I-attributes in anonymized data contradicts even the most basic definitions of privacy. Stopping now is a valid option comparable to pseudonymization techniques. However, as discussed in the introduction, it is not a good idea to do so. It is common knowledge, see [30, 5] for reference, that the combination of seemingly harmless attribute values can lead to identification and attribute disclosure. To respond to this problem, many authors in the area of k-anonymity categorize non-identifying attributes in quasi-identifiers and confidential attributes. Only QIs are considered to be able to identify individuals when combined. Typical examples are biological gender, date of birth or current town of residence. Unlike QIs, CAs contain sensitive information like income or severe diseases. However, as these attributes are not commonly known by someone trying to match a database entry to an individual, we do not consider them quasi-identifiable. According to this reasoning anonymization should provide privacy by deleting I-attributes and by altering QI-attributes in some way, to protect leakage of CA-attributes.

In my opinion the idea of separating between QI and CA is critically flawed. In most cases it is not clear whether an attribute is, or might become confidential in the future. Take for example the attribute *profession*. While in most cases someone's profession is not considered confidential, for some it might be. Another problem is the implicit assumption about the attacker's knowledge when classifying an attribute as CA. Is it safe to assume, the attribute *income* does not help identifying individuals? Further, what happens when an attribute is clearly confidential and quasi-identifiable at the same time? A solution I would like to suggest is to categorize every non-I-attribute as QI or in other words: do not separate non-I attributes. When anonymization is done right, there is no identity disclosure and hence confidential attributes are protected, even when classified as QI. Of course this model does not fit any scenario. We will therefore take a closer look at separation into QI

and CA in the context of alternative syntactic anonymity guarantees in section 3.3. For the main part of this text we will stick to my suggestion, concretely:

Remark 2.3.
For the purpose of anonymization, a database X is considered to contain quasi-identifiers only.

2.2 Anonymization

The anonymity of a database is a syntactic property. It guarantees that for every individual present in the data, there are at least $k-1$ other individuals with identical attribute values. The reasoning behind this definition is a concept called *hiding in a group of* k [66, 40]. It is assumed, k-anonymous data guarantees anonymity in the presence of an attacker with specific background knowledge, capabilities and identification goals.

Definition 2.4 (k-Anonymity [59, 66]).
A database X is *k-anonymous* if and only if for all $x \in \mathbb{R}^d$ it holds $|X_x| \geq k$ or $X_x = \emptyset$.

A common scenario for k-anonymity preventing privacy loss of an individual contained in public data is the stereotypical curious neighbor, asking themself why their target is admitted to hospital: It is reasonable to assume, the neighbor knows quite a lot about the target, so in a pseudonymized release of hospital records it would be easy to identify the record belonging to the target and hence obtaining information about medical conditions of their neighbor. If, however, the information is anonymized to be e.g. 10-anonymous prior to release, the best our attacker can hope for is to find the right group of 10 or more individuals in the data, which might or might not have similar attributes to their target. Even if the right group is found, the goal of determining the medical condition of the target cannot be achieved with certainty, because the respective attribute is also hidden in a group of 10 and might be generalized, aggregated or suppressed in the data released.

When talking about anonymity guarantees like k-anonymity or differential privacy there is a strong focus on the resulting data. But as our previous example shows, the process of anonymization has a big impact on how well anonymity and data utility is preserved when comparing two k-anonymous databases created out of the same original data using different anonymization techniques. To get a deeper understanding of the methods used to transform personally identifiable information into anonymized data it is necessary to define a precise model of the anonymization process itself.

The main focus of this text is *size conserving anonymization*. This term describes all forms of anonymization that alter attributes, but do not delete or change the order of the elements in the database. This setting originates from clustering algorithms and is used for all microaggregation algorithms.

Definition 2.5 (Size Conserving Anonymization).
A *size conserving anonymization (SCA) algorithm* $\mu_k : \mathcal{X}_{n\times d} \to \mathcal{X}_{n\times d}$ is a deterministic or probabilistic function parameterized with k, that guarantees $\mu_k(X) = \hat{X}$ is k-anonymous.

Determining the right measure of quality of a size conserving anonymization algorithm is non-trivial. Although anonymity is already covered by guaranteeing k-anonymity, the remaining utility of the resulting data can vary substantially from algorithm to algorithm. Hence, we should measure quality by comparing utility between algorithms. A natural way to do this would be to look at the research targets of the data recipient and to measure utility accordingly. However, the strength of k-anonymous data release is its universality: The concept of k-anonymity is designed to enable the same kinds of analyses as regular non-anonymized data offers. We can therefore not assume a specific research goal considering utility.

Fortunately, size conserving anonymization offers a valid alternative: It is possible to measure the distance between original and anonymized data in a meaningful way. We will now take a look at a general framework that is applicable for any kind of size conserving anonymization algorithm. In the next section, introducing the concept of microaggregation, the framework is specified to match the utility measurement found in the relevant literature.

Definition 2.6 (Distance).
Let $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ be a distance function in $\mathbb{R}^d$, $f : \mathbb{R}_+^n \to \mathbb{R}_+$ an aggregation function on sequences, x an element and finally $X = x_1, \ldots, x_n$ and $X' = x'_1, \ldots, x'_n$ two sequences of elements of the same length.

1. The *distance between two elements* x and x' is denoted $dist(x, x')$.
2. The *distance between two sequences* X and X' is $dist(X, X') := f((dist(x_i, x'_i))_{i\in\{1,\ldots,n\}})$.
3. The *distance between an element* x *and a sequence* X is $dist(x, X) := f((dist(x, x_i))_{i\in\{1,\ldots,n\}})$.

Depending on the use-case of the anonymization, several choices for $dist$ and f are possible. We could for example combine L_1 distance with a sum aggregation or L_2 distance with a max aggregation. Independently of these choices we define the distortion of a size conserving anonymization algorithm on a database as follows.

Definition 2.7 (SCA-Distortion).
Given a size conserving anonymization algorithm μ_k for any parameter k and a database X, the *SCA-distortion* of μ_k on X is $D_{\mu_k}(X) := dist(X, \mu_k(X))$.

In some cases, especially in the context of combining k-anonymity with differential privacy, it might be necessary or beneficial to reduce the size of the database to obtain valid anonymity guarantees. To capture this scenario the setting *size reducing anonymization* is used.

Definition 2.8 (Size Reducing Anonymization).
A *size reducing anonymization (SRA) algorithm* $\mu_k : \mathcal{X}_{n\times d} \to \mathcal{X}_{n'\times d}$ for $n' \leq n$ is a deterministic or probabilistic function parameterized with k which guarantees that $\mu_k(X) = \hat{X}$ is k-anonymous.

In cases in which the number or order of elements is changed during anonymization, a direct pre-post comparison is not possible. Nevertheless, general-purpose utility measures are achievable. An example for the use of size reducing anonymization is [40] where size reducing anonymization is used to achieve differential private data publishing without

the addition of noise, defining a new category of utility. Another usage is the *smallDB* mechanism [24], which defines utility for size reduced databases as the maximum distance between the answers to queries on the original and the reduced databases and hence defines a semantic utility measure under the assumption that a class of typical queries is known.

2.3 Microaggregation

Microaggregation is a technique achieving anonymity through clustering and is the main theme of this text. It has been introduced by Anwar, Defays and Nanopoulos [3, 15] in 1993 and predates the formal notion of k-anonymity from Sweeney by about five years. A first survey paper has been published in 1998 [51]. Similar to k-anonymity, microaggregation algorithms work on databases with one- or higher-dimensional data elements. Further, both rely on an anonymity factor k which describes the anonymity of the result. In contrast to k-anonymity, microaggregation focuses on the process of anonymization rather than the result itself and can be considered a technique to achieve k-anonymity. Elements are partitioned into clusters of size at least k and then each element is replaced by a representative of its cluster. As a result, microaggregated data is guaranteed to be k-anonymous.

At this point the following question might arise: What is offered by other k-anonymity methods that cannot be achieved through microaggregation? The answer to that is generality. Because microaggregation relies on clustering, the data has to be drawn from partially ordered sets to find a meaningful clustering and works best on metric spaces, where distance measures can be used. Further, microaggregation is limited to clustering approaches neglecting the possibilities offered by global suppression or generalization techniques. Other k-anonymity algorithms might not have these limitations and could, in theory, allow anonymization techniques that are applicable in more situations. Despite the similarities, both research fields are quite active until today, and techniques have been developed to overcome many shortcomings of both approaches. For example, there are techniques that extend the applicability of microaggregation onto categorical data [72, 50] and both, microaggregation and k-anonymity have been combined with differential privacy [40, 64]. Alternative k-anonymity approaches are discussed in chapter 3.

Before microaggregation can be defined formally, we need to clarify what exactly is meant by *partitioning into clusters of size at least k*. In difference to the more common k-clustering in which the data is split into k clusters of arbitrary size, a k-member clustering partitions the data into an arbitrary number of clusters, each with at least k elements.

Definition 2.9 (k-Clustering and k-Member Clustering).
Given a database $X \in \mathcal{X}_{n\times d}$ and an integer $k \geq 2$, a *clustering* can be described by a function

$$\mathcal{C}_X : \{1, \ldots, n\} \to \{1, \ldots, \ell\}$$

defining clusters $C_1, \ldots, C_\ell$ and assigning elements to clusters referring to their respective indices. When $\mathcal{C}_X(i) = j$, the element x_i is assigned to the cluster C_j. Slightly abusing this notation, we also write $C(x_i)$ and $x_i \in C_j$ referring to the cluster of an element x_i and the

elements of a cluster C_j, respectively. Whenever X is unambiguous we write $\mathcal{C}$ instead of $\mathcal{C}_X$.

A clustering $\mathcal{C}$ is a *k-clustering* iff $\ell = k$, i.e. there are exactly k (potentially empty) clusters.

A clustering $\mathcal{C}$ is a *k-member clustering* iff $|C_j| \geq k$ for all $1 \leq j \leq \ell$, i.e. any cluster has at least k elements.

General microaggregation is a size conserving anonymization built around a k-member clustering, defining clusters and building an anonymized database out of original data. Each element is represented and replaced by a cluster representative, which in combination with a k-member clustering leads to k-anonymity.

Definition 2.10 (General Microaggregation).
Microaggregation is a three-step process.

1. k-member clustering: A database X is partitioned into ℓ clusters $C_1, \ldots, C_\ell$ with $|C_j| \geq k$ for all $1 \leq j \leq \ell$.
2. Choosing representatives: A representative $y_j \in \mathbb{R}^d$ is chosen for every cluster C_j.
3. Replacing: All elements x_i of the database are replaced by a representative y_j of their cluster $C_j = C(x_i)$.

Because microaggregation is a size conserving anonymization scheme, distances and aggregation are not only used in the clustering step, but also for evaluation of the SCA-distortion. Throughout the literature the squared Euclidean distance $\delta(\cdot,\cdot)^2$ is used for *dist* and the sum aggregation is used for f for both clustering and utility measurement. Further the cluster representatives are chosen as centroids.

Definition 2.11 (Cluster Centroid).
Given a cluster C of elements x_i. The *centroid* of C is

$$c(C) := \frac{1}{|C|} \sum_{x_i \in C} x_i.$$

General microaggregation algorithms can either be *fixed-size* or *variable-size* also called *strict* or *non-strict* respectively. For fixed-size microaggregation all clusters are required to have exactly k elements, where $n/k \notin \mathbb{N}$ only one cluster is allowed to be bigger. Variable-size algorithms on the other hand do not further limit the clustering. As there is no benefit of fixed-size microaggregation when using it for the concept of hiding in a group of k, most modern algorithms use the greater possibilities of variable-size microaggregation to obtain better utility without the need to decrease k [61, 58, 63].

As described above, databases are modeled as sequences. This implies the possibility of duplicate elements. Whenever this happens a distinction between multiset-respecting and non-multiset-respecting microaggregation can be made. See table 2.2 for an example involving *TestData* from above. As for fixed-size microaggregation, the multiset-respecting property limits the choice of allowed clusters. It requires that all elements with identical data attributes are clustered together, potentially harming utility and preventing fixed-size microaggregation. Using the preferred choices of the relevant literature we obtain a specific type of microaggregation called *common microaggregation*.

Table 2.2: Two 3-anonymous databases created by microaggregation of the *TestData* database (table 2.1). Both databases use centroids as cluster representatives, but as x_3 and x_4 have identical attribute tuples in *TestData*, the clustering on the left is multiset-respecting while the clustering on the right is not.

Element	Attribute 1	Attribute 2
x_1	9	7
x_2	8	3
x_3	8	3
x_4	8	3
x_5	9	7
x_6	9	7

(a) multiset-respecting

Element	Attribute 1	Attribute 2
x_1	6	4
x_2	6	4
x_3	6	4
x_4	11	6
x_5	11	6
x_6	11	6

(b) non-multiset-respecting

Definition 2.12 (Common Microaggregation).
Common microaggregation describes a subset of the set of all possible general microaggregation algorithms.

1. k-member clustering: A database X is partitioned into ℓ clusters $C_1, \ldots, C_\ell$ with $|C_j| \geq k$ for all $1 \leq j \leq \ell$. The clustering may be non-strict and non-multiset-respecting.
2. Centroid calculation: A representative $y_j \in \mathbb{R}^d$ is chosen as the centroid $c(C_j)$ for every cluster C_j.
3. Replacing: All elements x_i of the database are replaced by the representative y_j of their cluster $C_j = C(x_i)$.

Instead of just looking at the global SCA-distortion of a microaggregation algorithm, the choices of common microaggregation allow us to obtain much more detailed information by considering the cost of particular clusters. Please note that most of the results below also apply for general microaggregation algorithms that use cluster centroids as representative i.e. strict and/or multiset-respecting common microaggregation algorithms.

Definition 2.13 (Distortion Cost).
Given a database X clustered into a set of clusters $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ by a common microaggregation algorithm μ_k (or any other clustering algorithm).

– Consider two tuples $x = (x^1, \ldots, x^d), y = (y^1, \ldots, y^d) \in \mathbb{R}^d$. The *Euclidean distance* between x and y is

$$\delta(x, y) := \sqrt{\sum_{i=1}^{d} (x^i - y^i)^2}.$$

The *squared Euclidean distance* between x and y is

$$\delta(x, y)^2 = \sum_{i=1}^{d} (x^i - y^i)^2.$$

- The *cluster cost* or *cost of a cluster* C_j is the intra-cluster sum of squared errors (*SSE*) of C_j denoted and computed as $cost(C_j) = SSE(C_j) := \sum_{x \in C_j} \delta(x, c_j)^2$.
- The *distortion cost* of μ_k on X is the sum of all cluster costs: $cost(\mathcal{X}, \mu_k) = cost(\mathcal{C}) := \sum_{C_j \in \mathcal{C}} cost(C_j)$.

As all elements of a cluster are replaced by the cluster centroid in step 3 of the microaggregation process, we obtain the cost of a cluster by calculating the intra-cluster sum of squared errors. Adding up all cluster costs is equal to the distance $dist(X, \mu_k(X)) = \sum_{x_i \in X} \delta(x_i, C(x_i))^2$ and hence the SCA-distortion of a common microaggregation algorithm is equal to the distortion cost described in definition 2.13. The question might arise whether the choice of a centroid as cluster representative is optimal. A simple calculation can confirm this hypothesis.

Theorem 2.14.
Consider any cluster C_j. $SSE(C_j) = \sum_{x \in C_j} \delta(x, c_j)^2$ is minimal for $c_j = 1/|C_j| \sum_{x \in C_j} x$.

Proof.

$$SSE(C_j) = \sum_{x \in C_j} \delta(x, c_j)^2 = \sum_{x \in C_j} (x - c_j)^2 = \sum_{x \in C_j} (x^2 - 2xc_j + c_j^2).$$

By finding the root of the partial derivate

$$\frac{\partial\ SSE(C_j)}{\partial\ c_j} = \sum_{x \in C_j} (2c_j - 2x) = 2c_j|C_j| - 2\sum_{x \in C_j} x$$

we can see that the centroid

$$c_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

indeed minimizes the sum of squared errors. □

Using a result from the research field of k-means clustering one can show even more: By using the cluster size and distance of the representative to the cluster centroid it is possible to calculate the increase in cluster cost caused by using a different representative than the centroid. Theorem 2.15 is used but not proven in [4].

Theorem 2.15.
Let C_j be a cluster with centroid c_j and z be an arbitrary (existing or hypothetical) element within or not within C_j. Then

$$\sum_{x \in C_j} \delta(x, z)^2 - \sum_{x \in C_j} \delta(x, c_j)^2 = |C_j| \cdot \delta(c_j, z)^2.$$

Proof.

$$\begin{aligned}
&\sum_{x\in C_j}\delta(x,z)^2-\sum_{x\in C_j}\delta(x,c_j)^2=\sum_{x\in C_j}(x^2-2xz+z^2)-\sum_{x\in C_j}(x^2-2xc_j+c_j^2)\\
=&\sum_{x\in C_j}(x^2-2xz+z^2)-\sum_{x\in C_j}\left(x^2-2x\cdot\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)+\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)^2\right)\\
=&\sum_{x\in C_j}(-2xz+z^2)-\sum_{x\in C_j}\left(-2x\cdot\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)+\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)^2\right)\\
=&-2z\sum_{x\in C_j}x+|C_j|z^2+2\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)\cdot\sum_{x\in C_j}x-|C_j|\cdot\left(\frac{\sum_{y\in C_j}y}{|C_j|}\right)^2\\
=&-2z\sum_{x\in C_j}x+|C_j|z^2+\frac{2\sum_{x\in C_j}x}{|C_j|}\cdot\sum_{x\in C_j}x-|C_j|\cdot\frac{\sum_{x\in C_j}x^2}{|C_j|^2}\\
=&-2z\sum_{x\in C_j}x+|C_j|z^2+\frac{2\sum_{x\in C_j}x^2}{|C_j|}-\frac{\sum_{x\in C_j}x^2}{|C_j|}\\
\\
=&-2z\sum_{x\in C_j}x+|C_j|z^2+\frac{\sum_{x\in C_j}x^2}{|C_j|}\\
=&|C_j|\cdot\left(\left(\frac{\sum_{x\in C_j}x}{|C_j|}\right)^2-2z\left(\frac{\sum_{x\in C_j}x}{|C_j|}\right)+z^2\right)\\
=&|C_j|\cdot\left(c_j^2-2zc_j+z^2\right)=|C_j|\cdot\delta(c_j,z)^2
\end{aligned}$$

□

As a consequence of theorem 2.14 it is easy to see that step 2 and 3 of definition 2.12 are simple deterministic computations whereas the complexity of microaggregation solely depends on the implementation of step 1. Hence the formal microaggregation problems used extensively in chapter 4 are defined as follows.

Definition 2.16 (Optimal k-Anonymous Microaggregation Problem).
Given a database X, the *optimal k-anonymous microaggregation problem* is to find a k-member clustering $\mathcal{C}$ for X with minimum distortion cost.

Definition 2.17 (Decisional k-Anonymous Microaggregation Problem).
Given a database X and a number $\gamma\in\mathbb{R}$, the *decisional k-anonymous microaggregation problem* is to determine whether a k-member clustering $\mathcal{C}$ with distortion cost at most γ exists for X.

2.4 Elementary Results Regarding Cost Computations

The squared Euclidean distance is commonly used for the computation of distances in clustering problems. This is mostly because it offers some unique features like a linear dependency of distances in different dimensions and other useful properties described in this section. However, it should be noted that squared Euclidean distance does not form a metric space, as the triangle inequality does not hold. A simple example for this is the set of elements $\{x_1 = (0,0), x_2 = (2,1), x_3 = (4,0)\}$ for which $\delta(x_1, x_2)^2 = \delta(x_2, x_3)^2 = 5$ but $\delta(x_1, x_3)^2 = 16$.

In some situations, especially for experimenting with microaggregation techniques on paper, it might be desirable to calculate cluster cost using distances between elements and without determining the cluster centroids first. The following result confirms the existence of such an alternative computation method.

Theorem 2.18.
Consider any cluster C_j. The cost of C_j can be expressed as

$$cost(C_j) = \frac{1}{|C_j|} \sum_{\{x_a, x_b\} \subseteq C_j} \delta(x_a, x_b)^2 = \frac{1}{2|C_j|} \sum_{(x_a, x_b) \in C_j \times C_j} \delta(x_a, x_b)^2.$$

Proof. Let $s := |C_j|$, $c := c(C_j)$ and x_1 to x_s be the elements of C_j.

$$\frac{1}{2s} \sum_{(x_a, x_b) \in C_j} \delta(x_a, x_b)^2 = \frac{1}{2s} \begin{pmatrix} (x_1 - x_2)^2 + \cdots + (x_1 - x_s)^2 + \\ (x_2 - x_1)^2 + \cdots + (x_2 - x_s)^2 + \\ \cdots + \\ (x_s - x_1)^2 + \cdots + (x_s - x_{s-1})^2 \end{pmatrix}$$

$$= \frac{1}{2s} \left(\sum_{a=1}^{s} (s-1) x_a^2 + \sum_{a=1}^{s} (s-1) x_a^2 - \sum_{a=1}^{s} \sum_{\substack{b=1 \\ b \neq a}}^{s} 2 x_a x_b \right)$$

$$= \frac{1}{2s} \left(2(s-1) \sum_{a=1}^{s} x_a^2 - 2 \sum_{a=1}^{s} \sum_{\substack{b=1 \\ b \neq a}}^{s} x_a x_b \right)$$

$$= \frac{s-1}{s} \sum_{a=1}^{s} x_a^2 - \frac{1}{s} \sum_{a=1}^{s} \sum_{\substack{b=1 \\ b \neq a}}^{s} x_a x_b \tag{2.1}$$

$$
\begin{aligned}
&cost(C_j) = \sum_{a=1}^{s} \delta(x_a, c)^2 \\
=&(x_1 - c)^2 + (x_2 - c)^2 + \cdots + (x_s - c)^2 \\
=&\sum_{a=1}^{s} x_a^2 + sc^2 - 2c \sum_{a=1}^{s} x_a = \sum_{a=1}^{s} x_a^2 + sc^2 - 2c \cdot sc = \sum_{a=1}^{s} x_a^2 - sc^2 \\
=&\sum_{a=1}^{s} x_a^2 - \frac{1}{s} \left(\sum_{a=1}^{s} x_a \right)^2 = \sum_{a=1}^{s} x_a^2 - \frac{1}{s} \left(\sum_{a=1}^{s} x_a^2 + 2 \cdot \sum_{a=1}^{s} \sum_{b=a+1}^{s} x_a x_b \right) \\
=&\frac{s-1}{s} \sum_{a=1}^{s} x_a^2 - \frac{2}{s} \sum_{a=1}^{s} \sum_{b=a+1}^{s} x_a x_b
\end{aligned}
$$

$$
= \frac{s-1}{s} \sum_{a=1}^{s} x_a^2 - \frac{1}{s} \sum_{a=1}^{s} \sum_{\substack{b=1 \\ b \neq a}}^{s} x_a x_b \tag{2.2}
$$

Since term 2.1 and term 2.2 are equal, the alternative computation method is correct. □

Two natural operations on clusterings are the union of two clusters and the incorporation of an additional element into an existing cluster. Theorem 2.19 and corollary 2.20 provide useful guarantees regarding cluster cost in such situations. By applying theorem 2.15 we can precisely calculate the additional cost emerging from a set union. Using this property in algorithms could potentially reduce computational complexity, as the cost of new clusters might not need to be computed from scratch.

Theorem 2.19.
The union of any two clusters C_1, C_2 of a clustering $\mathcal{C}$ cannot have cost lower than the sum of costs for both clusters alone:

$$cost(C_1 \cup C_2) \geq cost(C_1) + cost(C_2),$$

more precisely:

$$
\begin{aligned}
cost(C_1 \cup C_2) =& cost(C_1) + cost(C_2) \\
&+ |C_1| \cdot \delta(c(C_1), c(C_1 \cup C_2))^2 \\
&+ |C_2| \cdot \delta(c(C_2), c(C_1 \cup C_2))^2.
\end{aligned}
$$

Proof. Let $s_1 := |C_1|, s_2 := |C_2|, c_1 := c(C_1), c_2 := c(C_2)$ and $c_U := c(C_1 \cup C_2)$.

$$
\begin{aligned}
& cost(C_1 \cup C_2) \\
=& \sum_{x \in C_1 \cup C_2} \delta(x, c_U)^2 \\
=& \sum_{x \in C_1} \delta(x, c_U)^2 + \sum_{x \in C_2} \delta(x, c_U)^2 && (C_1, C_2 \text{ disjoint}) \\
=& \sum_{x \in C_1} \delta(x, c_1)^2 + s_1 \cdot \delta(c_1, c_U)^2 + \sum_{x \in C_2} \delta(x, c_2)^2 + s_2 \cdot \delta(c_2, c_U)^2 && (\text{Theorem 2.15}) \\
=& cost(C_1) + cost(C_2) + s_1 \cdot \delta(c_1, c_U)^2 + s_2 \cdot \delta(c_2, c_U)^2 \\
\geq& cost(C_1) + cost(C_2)
\end{aligned}
$$

□

Corollary 2.20.
Adding an element x to a cluster C_j cannot decrease its cost:

$$cost(C_j \cup x) \geq cost(C_j),$$

more precisely:

$$cost(C_j \cup x) = cost(C_j) + |C_j| \cdot \delta(c(C_j), c(C_j \cup x))^2 + \delta(x, c(C_j \cup x))^2.$$

Corollary 2.21 is especially useful proving complexity results and optimality guarantees for microaggregation algorithms.

Corollary 2.21.
If a clustering contains a cluster C_j of size at least $2k$, C_j can always be split into two clusters each of size at least k without increasing distortion cost of the resulting k-member clustering.

The next corollary shows that we can obtain the centroid of a merged cluster without any distance computations. In combination with theorem 2.19 this allows further improvement in the update complexity of iterative clustering algorithms.

Corollary 2.22.
The centroid of the union of any two clusters C_1, C_2 can be computed from just sizes and centroids of C_1 and C_2.

Proof. Let $C := C_1 \cup C_2$, $c_1 := c(C_1), c_2 := c(C_2), n_1 = |C_1|$ and $n_2 = |C_2|$. The centroid of C is $c(C) = \frac{c_1 \cdot n_1}{n_1 + n_2} + \frac{c_2 \cdot n_2}{n_1 + n_2}$. □

Comparing distortion costs is particularly useful when several algorithms are benchmarked against each other on the same database. However, as these costs are not normalized, it is not possible to evaluate the effect of one microaggregation algorithm on different databases just comparing distortion. For this purpose the scientific community around microaggregation introduced the notion of *information loss*. The distortion cost is divided by a database-specific *diversity* resulting in a value ranging between 0 and 1.

Definition 2.23 (Information Loss).
The *diversity* $\Delta(X)$ of a database X is the total sum of squares between all elements $x_i \in X$ and a global centroid c:

$$\Delta(X) = SST(X) := \sum_{i=1}^{n} \delta(x_i, c)^2.$$

The *information loss* $L(X, \mu_k)$ of a common microaggregation algorithm μ_k on a database X is

$$L(X, \mu_k) := \frac{cost(X, \mu_k)}{\Delta(X)}.$$

As fact 2.24 shows, the information loss is effectively a normalized distortion cost of a microaggregation algorithm μ on a database X. Because of this one often sees the value multiplied by 100 and hence expressed as an percentage of information loss (see e.g. [61, 19]).

Fact 2.24.
Let X be a database and μ_k a common microaggregation algorithm, clustering X into a set of clusters $\mathcal{C} = \{C_1, \ldots, C_l\}$. If X contains at least two different elements, i.e. $\exists x_1, x_2 : |X_{x_1}| \geq 1, |X_{x_2}| \geq 1$ and $X_{x_1} \neq X_{x_2}$ we obtain

$$0 \leq L(X, \mu_k) \leq 1,$$

(otherwise $L(X, \mu_k) = 0/0$).

Proof. The inequality $0 \leq L(X, \mu_k) = cost(X, \mu_k)/\Delta(X)$ holds, because $cost(X, \mu_k)$ is always non-negative and $\Delta(X)$ is always positive.

For $L(X, \mu_k) \leq 1$ we show that $cost(X, \mu_k) = \sum_{C_j \in \mathcal{C}} \sum_{x \in C_j} \delta(x, c(C_j))^2 \leq \Delta(X)$ holds for any partition of X into a clustering $\mathcal{C}$: As $\Delta(X)$ is effectively the cost of the union of all clusters, applying theorem 2.19 shows that no clustering can have a cost higher than $\Delta(X)$. □

3

Microaggregation in a broader context

The definitions of databases (definition 2.1), k-anonymity (definition 2.4) and size conserving anonymization (definition 2.5) are tailored towards microaggregation which relies on numeric data and the computability of distances between elements. There are, however, related methods that also use clustering to achieve k-anonymity but are clearly distinct from microaggregation. This chapter is dedicated to these alternative approaches, their data, utility and anonymity units.

The closest relative to microaggregation in the field of k-anonymity is the so-called recoding scheme (e.g. [74]). While there are few links between both research fields, the similarity of both approaches allows us to consider microaggregation in a broader context and explore results that would apply if we alter our definitions slightly. The general idea of a recoding scheme is to transform original d-dimensional database elements into new elements drawn from a potentially different set of d dimensions. This allows for example to generalize an attribute of an element to a range of possible attributes, or suppressing some attributes entirely. Both techniques contribute to the uniformity of the data and as a result work towards achieving k-anonymity.

There are two types of recoding schemes. A global recoding scheme partitions a database $X \in (\mathcal{D}_1 \times \cdots \times \mathcal{D}_d)^n$ into clusters C_j and changes the elements' attributes to some statistics about all elements of their cluster. In contrast to microaggregation these statistics do not need to be in the form of single representative vectors but instead could be e.g. information about the boundaries of clusters or the information that some attribute values have been suppressed. A global recoding scheme is always multiset-respecting as the clustering is based on the attributes of elements and no two identical elements could end up in two different clusters.

Definition 3.1 (Global Recoding Scheme).
A *global recoding scheme* is a three-step process.

1. k-member clustering: A database $X \in (\mathcal{D}_1 \times \cdots \times \mathcal{D}_d)^n$ is partitioned into ℓ **multiset-respecting** clusters $C_1, \ldots, C_\ell$ with $|C_j| \geq k$ for all $1 \leq j \leq \ell$.
2. Cluster statistics $y_j = (y_j^1, \ldots, y_j^d) \in \mathcal{D}'_1 \times \cdots \times \mathcal{D}'_d$ are computed for every cluster C_j.
3. Replacing: All elements x_i of the database are replaced by the cluster statistics Y_j of their cluster $C_j = C(x_i)$ creating a new database $Y \in (\mathcal{D}'_1 \times \cdots \times \mathcal{D}'_d)^n$.

A local recoding scheme is more general as it allows cluster assignment on a record by

record basis instead of clustering by attribute vectors. Clusters might overlap and hence local recoding allows non-multiset-respecting clustering of data.

Definition 3.2 (Local Recoding Scheme).
A *local recoding scheme* is a global recoding scheme for which the k-member clustering in step 1 does not have to be multiset-respecting:

1′. k-member clustering: A database $X \in (\mathcal{D}_1 \times \cdots \times \mathcal{D}_d)^n$ is partitioned into ℓ clusters $C_1, \ldots, C_\ell$ with $|C_j| \geq k$ for all $1 \leq j \leq \ell$.

Note that suppression is achieved by allowing the empty string $\varepsilon \in \mathcal{D}'_i$ and that the original and resulting databases in definition 3.1 and definition 3.2 are not necessarily databases according to definition 2.1, as the attribute domain of X and Y might not be $\mathbb{R}^d$. See section 3.1 for generalized definitions of databases and k-anonymity. As a third note I would like to emphasize that a local recoding scheme as defined in definition 3.2 is the most general clustering algorithm defined as a size conserving anonymization algorithm. It allows to replace any elements by arbitrary d-dimensional data, as long as k-anonymity is guaranteed in the end. Typically, however, local recoding schemes act like global recoding schemes with the option of duplicate data to be split in different clusters. An example for global and local recoding is given in table 3.1.

Any global recoding scheme is also a valid local recoding scheme. Further any general microaggregation algorithm can be interpreted as a local recoding scheme and any multiset-respecting general microaggregation algorithm can be interpreted as a global recoding scheme where representative vectors are used as cluster statistics.

Table 3.1: Two 3-anonymous datasets created by a generalizing recoding of the *TestData* database (table 2.1). As x_3 and x_4 have identical attribute tuples in *TestData*, the clustering on the left is a global recoding while the clustering on the right is not.

Element	Attribute 1	Attribute 2
x_1	$3-12$	$6-9$
x_2	$6-9$	3
x_3	$6-9$	3
x_4	$6-9$	3
x_5	$3-12$	$6-9$
x_6	$3-12$	$6-9$

(a) global recoding

Element	Attribute 1	Attribute 2
x_1	$3-9$	$3-6$
x_2	$3-9$	$3-6$
x_3	$3-9$	$3-6$
x_4	$9-12$	$3-9$
x_5	$9-12$	$3-9$
x_6	$9-12$	$3-9$

(b) local recoding

Many algorithms categorized as recoding schemes allow the data to be either numerical or categorical, as long as there is a partial order for every dimension. They therefore require slightly different definitions for databases and k-anonymity. In the first section of this chapter we take a look at non-numerical data and how we need to alter our definitions to enable us to apply microaggregation and recoding schemes in the presence of categorical data.

The measurement of utility is also effected by allowing a wider range of data before and after the anonymization. As microaggregation algorithms can be expressed as a constrained

type of recoding schemes, several utility measures designed with recoding schemes in mind are also applicable for microaggregation. In the second section we will take a look at some of these guarantees and compare them with our standard definitions of distortion, diversity and information loss.

We conclude this chapter by taking a look at other syntactic anonymity guarantees related to k-anonymity. While they once have been promising possible successors to k-anonymity, most of these guarantees are now abandoned by the scientific community due to utility concerns and the rise of semantic privacy guarantees like differential privacy which promise to offer superior privacy with reasonable amounts of utility. Nevertheless, as this thesis mainly covers results regarding k-anonymity, taking a closer look at the limits and broader context of this aspect is needed as well.

3.1 Non-Numerical Data

Most real-world datasets contain non-numerical data. There are (partially) ordered non-numerical attributes like *university degree* where one could agree that a master's degree is in a way *more* than a bachelor's degree, but there is no natural mean of e.g. two bachelor's- and three master's-degrees. Further, even if there is some sort of hierarchy given, this still does not mean, that one can compute distances between attribute values: What is the distance between a master's degree and a Ph.D.? Is it greater than the distance between a master's degree and a bachelor's degree? If so, how much greater is it? There also exist unordered non-numerical attributes like profession, where there is no clear relationship or distance measure between different attribute values like e.g. a dentist and a farmer.

Most recoding schemes allow categorical data only in the presence of user-defined generalization hierarchies, transforming unordered into partially ordered non-numerical data (see e.g. [37, 75]). We will therefore concentrate on this scenario exclusively.

To incorporate non-numerical data, we need to alter a few of our definitions from chapter 2 to allow for that change. To avoid any confusion about when to apply which definition, please assume that any reference to databases and k-anonymity outside of this chapter always uses the original definition from chapter 2. At first, a reformulation of databases is needed, as our original definition does not allow elements outside of $\mathbb{R}^d$.

Definition 3.3 (Extension of Definition 2.1 to Include Non-Numerical Data).
A *database element* $x_i = (x_i^1, \ldots, x_i^d)$ is a d-dimensional tuple of data drawn from a set $\mathcal{D}_1 \times \cdots \times \mathcal{D}_d$. A *database* is a sequence $X = x_1, \ldots, x_n$ of database elements, potentially including duplicate elements. The set of all possible databases of n elements in dimensions $\mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ is $\mathcal{D}_{n\times d} := (\mathcal{D}_1 \times \cdots \times \mathcal{D}_d)^n$.

Another subtle change is needed in the definition of k-anonymity. While the general statement is still the same, we now allow non-numerical databases.

Definition 3.4 (Alternative Formulation of Definition 2.4 to Include Non-Numerical Data).
A database $X \in \mathcal{D}_{n\times d}$ is *k-anonymous* if and only if for all $x \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ it holds $|X_x| \geq k$ or $X_x = \emptyset$.

As stated before, microaggregation in its basic form described above is not able to handle non-numerical data. Recoding schemes on the other hand are. However, to allow a cluster to get assigned cluster statistics, we need the right tool. This tool is called *hierarchy trees*. Hierarchy trees can be applied to any kind of numerical or categorical data and define groups which dictate the data basis for cluster statistics.

Definition 3.5 (Hierarchy Trees).
An *attribute hierarchy tree for (categorical or numerical) attributes from a set* $\mathcal{D}_j$ is a tuple $H_j = (N_j, p_j)$ consisting of a set of nodes N_j, representing sets of attribute values $d \in \mathcal{D}_j$ and a parenting function p_j assigning a parent node $p_j(n)$ to each node $n \in N_j$. Leaf nodes (nodes which are not parent to any other node) consist of sets of single attribute values. Every attribute value $d \in \mathcal{D}_j$ has a leaf node representing it. The root node $r_j \in N_j$ contains all values from $\mathcal{D}_j$, and $p_j(r_j) = r_j$. The tree does not have to be binary and has no restrictions regarding depth or balance. For numerical attributes, there might be indefinitely many nodes.
A *tuple of attribute hierarchy trees* $H(\mathcal{D}_{n\times d}) = (H_1, \ldots, H_d)$ for a d-dimensional attribute space $\mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ defines generalization hierarchies for databases $X \in \mathcal{D}_{n\times d}$. The set of all possible hierarchy trees for databases $X \in \mathcal{D}_{n\times d}$ is denoted by $\mathcal{H}(\mathcal{D}_{n\times d})$.

For a hierarchy tree H_j the *size of a node* n defined by a function $size_{H_j} : N_j \to \mathbb{N}$ describes the number of leaves which have the node n as an ancestor. The size of any leaf is 0 and the size of r_j is $|\mathcal{D}_j|$.

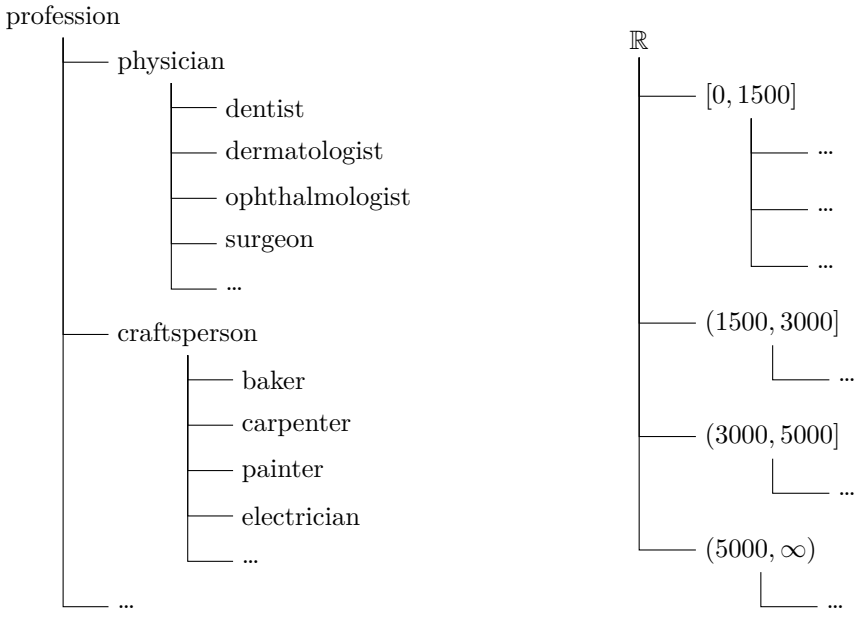
The *closest common ancestor* in a hierarchy tree H_j is defined on a set of nodes $N' \subseteq N_j$ and is written as $cca_{H_j}(N')$. The closest common ancestor of a single node is the node itself.

Hierarchy trees are especially useful for algorithms based on the most common recoding paradigm: *generalization*. Generalization is an anonymization technique used in situations in which distances between attributes cannot be computed but single values instead of ranges or sets should be used as attribute representatives. Hence, it is often associated with non-numerical data, but is compatible with numerical data as well.

Definition 3.6 (Generalization and Suppression).
A *generalization algorithm* $\mu_k : \mathcal{D}_{n\times d} \times \mathcal{H}(\mathcal{D}_{n\times d}) \to \mathcal{D}'_{n\times d}$ is a local recoding scheme in which the cluster statistics are computed based on the closest common ancestors of the cluster elements in the respecting hierarchy trees.

A generalization algorithm uses *suppression* if it outputs $y_i^j = \varepsilon$ as cluster statistics of a dimension j of any cluster C_i. A generalization algorithm using suppression is called *suppressing*.

Let's consider an example. Assume there are two attribute dimensions $\mathcal{D}_1$ and $\mathcal{D}_2$ describing the categorical attribute *profession* and the numerical attribute *monthly net salary in Euro*, respectively. Typical hierarchy trees H_1 and H_2 might look like depicted in figure 3.2. An exemplary database $X \in \mathcal{D}_{7\times 2}$ and a 2-anonymous generalization Y of it are given in table 3.3.



(a) H_1 for $\mathcal{D}_1$ (b) H_2 for $\mathcal{D}_2$

Figure 3.2: Exemplary hierarchy trees H_1 (a) and H_2 (b) for attribute sets $\mathcal{D}_1 = \{$*dentist, dermatologist, ophthalmologist, surgeon, baker, carpenter, painter, electrician, ...*$\}$ and $\mathcal{D}_2 = \mathbb{R}$.

Table 3.3: An original database X is generalized into a 2-anonymous database Y using hierarchy trees of figure 3.2 and suppression.

Element	Attribute 1	Attribute 2
x_1	dentist	2800
x_2	dentist	5800
x_3	baker	2200
x_4	painter	2900
x_5	painter	3137
x_6	surgeon	3137
x_7	electrician	3137

(a) original database $X \in \mathcal{D}_{7\times 2}$

Element	Attribute 1	Attribute 2
x_1	dentist	ϵ
x_2	dentist	ϵ
x_3	craftsperson	$(1500, 3000]$
x_4	craftsperson	$(1500, 3000]$
x_5	ϵ	3137
x_6	ϵ	3137
x_7	ϵ	3137

(b) generalized database $Y \in \mathcal{D}'_{7\times 2}$

3.2 Alternative Utility Units

Recoding schemes are not formally categorized as size conserving anonymization algorithms, because attributes of elements are replaced not by other attributes from the same attribute set, but by cluster statistics which potentially include values from other sets. Hence, applying SCA-distortion is not possible.

The community developed different utility measures applicable for recoding schemes. Most of these measures are applicable for microaggregation algorithms as well. The most basic utility measure thinkable is cluster size. There is a hard lower bound of k which allows the concept of hiding in a group of k. Further, as anonymity is evaluated in a worst case manner, having some clusters with more elements does not improve anonymity. Hence, one could argue that bigger clusters unnecessarily reduce utility and that statistics regarding cluster sizes of recoding algorithms somehow reflect quality of data. The *discernability metric* C_{DM} [7, 37] and *normalized average equivalence class size metric* C_{AVG} [37] are the most common variants of this approach.

Definition 3.7 (Utility Based on Cluster Size).
The *discernability metric* C_{DM} describes the sum of squared cluster sizes of a k-member clustering:

$$C_{DM} = \sum_{j=1}^{\ell} |C_j|^2$$

The *normalized average equivalence class size metric* C_{AVG} computes the average cluster size, by dividing the number of elements n by the number of clusters ℓ. The result is divided by the anonymity guarantee k.

$$C_{AVG} = \left(\frac{n}{\ell}\right) / k$$

The reasoning behind C_{DM} is that clusters are sanctioned over-linearly with growing size which emphasizes the negative effect on homogeneity of any additional element to a cluster. There is, however, a fundamental problem with this approach: Results between different databases or even different k on the same database are not comparable. Larger databases will inevitably result in a greater discernability. Further, as the *cost* of a cluster grows over-linearly, raising the minimum cluster size k effectively sanctions additional elements to clusters more. This does not only introduce problems with comparability, but is also counter-intuitive. As clusters are bigger anyway, we expect additional elements to do less harm instead of more.

All of these problems are remedied in the C_{AVG} metric. As average cluster size is concerned, databases of different sizes are evaluated equally. Further, normalizing the result by dividing by k eliminates the dependency on the minimum cluster size and sets the value $C_{AVG} = 1$ as the best case in any situation. The *penalty* of a clustering C can be expressed by $C_{AVG}(C) - C_{AVG}(OPT) = C_{AVG}(C) - 1$. Both clusters from table 3.1 have $C_{DM} = 18$ and $C_{AVG} = 1$ as both clusterings use the same cluster sizes.

One might ask whether cluster size actually accurately reflects utility. Of course the answer to this question depends on the application and especially on the type of cluster statistics that are computed, transforming a k-member clustering into a microaggregation or recoding algorithm. For microaggregation and generalization, cluster size is a rather weak indicator for utility. As cluster statistics are computed depending on the range of attribute values inside a cluster, a good utility measure should take the range of elements inside clusters into account. The *normalized certainty penalty* (NCP) introduced in [75] implements this approach.

The NCP is defined for numerical, categorical or mixed databases, allows weighting of certain attribute dimensions and considers distances between extreme attribute values inside of clusters. However, the density of clusters is not evaluated.

Definition 3.8 (Normalized Certainty Penalty [75]).
The *normalized certainty penalty* (NCP) of an element x_i is a weighted sum of penalties of all dimensions j of x_i:

$$NCP(x_i) = \sum_{j=1}^{d} \left(w_j \cdot NCP_{\mathcal{D}_j}(x_i^j) \right)$$

If x_i^j is numerical, $NCP_{\mathcal{D}_j}(x_i^j)$ is the difference between the largest and smallest attribute value in dimension j inside of cluster $C(x_i)$ divided by the difference between the largest and smallest attribute value in dimension j over the whole database:

$$NCP_{\mathcal{D}_j}(x_i^j) = \frac{max_j(C(x_i)) - min_j(C(x_i))}{max_j(X) - min_j(X)}$$

If x_i^j is categorical, $NCP_{\mathcal{D}_j}(x_i^j)$ considers the hierarchy tree H_j to determine the size of the closest common ancestor of all attribute values of elements in dimension j and cluster $C(x_i)$. This value is normalized by the total number of categories of $\mathcal{D}_j$ i.e. the number of leaves in H_j:

$$NCP_{\mathcal{D}_j}(x_i^j) = \frac{size_{H_j}(cca_{H_j}(C(x_i)))}{|\mathcal{D}_j|}$$

The normalized certainty penalty of a database X is

$$NCP(X) = \sum_{i=1}^{n} NCP(x_i) = \sum_{i=1}^{n} \sum_{j=1}^{d} \left(w_j \cdot NCP_{\mathcal{D}_j}(x_i^j) \right).$$

For both numerical and categorical attributes, an NCP of 1 is the worst case which occurs when a cluster contains the elements with maximal distance to each other. An NCP of 0 is achieved if only one attribute value occurs in a cluster. Again, this property holds for numerical as well as for categorical data.

In difference to C_{DM} and C_{AVG}, NCP evaluates both clusterings from table 3.1 differently. The anonymization of table 3.1a has an NCP of 5.5 and table 3.1b has an NCP of 7.5, assuming all attributes j have weight $w_j = 1$.

When NCP is used to compare the utility-preserving properties of generalization algorithms, the penalty accurately measures the loss of information. Generalization does not care for density and anonymizes based on extreme values inside of clusters only. Hence, for categorical data, NCP measures the exact sizes of generalized attributes in the resulting anonymized data. For numerical data, however, NCP might underestimate the cost, as it ignores the fact that there might not be an ancestor node for the exact range of attribute values inside a cluster. This rather technical problem can be fixed quite easily using one of the following approaches:

- For numerical attributes j only allow hierarchy trees H_j which have nodes for every subset of $\mathcal{D}_j$.
- Change NCP so that size of cca is used for numerical attributes as well.
- Allow generalization algorithms to generalize numerical attributes based on extreme values instead of hierarchy trees.

When it comes to other recoding algorithms beside generalization, NCP is less appealing. For microaggregation for example, the extreme values of attributes inside clusters provide little information about utility, as distortion of data is based only on density. I.e. in a dense cluster the difference between an original attribute value and an attribute value after replacement with the centroid is typically smaller than in a sparse cluster.

A slightly different notation of the same utility concept can be found under the name *total information loss* (Total-IL) in [12]. The main difference between NCP and Total-IL is that for categorical data, in Total-IL the height of the closest common ancestor in a hierarchy tree is divided by the total height of the hierarchy tree, whereas for NCP the number of leaves reachable from a closest common ancestor is counted and divided by the total number of leaves. Further, no attribute-weighting is used, although this could be implemented quite easily. As both approaches are very similar, we will not go into further details about Total-IL. For more information see [12].

3.3 Stronger Anonymity Guarantees

So far, the only anonymity guarantee considered was k-anonymity. We discussed new forms of databases and new approaches to measure utility. However, until now we did not look into different approaches of measuring anonymity. In the next few paragraphs we will take a look at some alternatives to k-anonymity which emerged between 2007 and 2014. All of these approaches try to improve anonymity guarantees given by k-anonymity by further restricting the choice of clusters by a recoding scheme. It should be noted right away that these constraints inevitably lead to a reduction in utility [11, 41]. For more information on other anonymity guarantees see [28].

Diversity and Closeness

As mentioned in chapter 2, most syntactic anonymity guarantees assume data to be divided into quasi-identifiers (QI) and confidential attributes (CA). Because most anonymity guarantees discussed in this section rely on a QI/CA separation, we first need to introduce some new notations regarding these types of attributes.

Definition 3.9 (Quasi-Identifiers and Confidential Attributes).
Let $X = x_1, \ldots, x_n$ be a database of n elements drawn from $\mathcal{D}_1 \times \cdots \times \mathcal{D}_d$.

- The sets $\mathcal{D}_{QI} \subseteq \{1, \ldots, d\}$ and $\mathcal{D}_{CA} \subseteq \{1, \ldots, d\}$ with $\mathcal{D}_{QI} \cap \mathcal{D}_{CA} = \emptyset$ and $\mathcal{D}_{QI} \cup \mathcal{D}_{CA} = \{1, \ldots, d\}$ are the *sets of QI-dimensions* and *CA-dimensions* respectively.
- The *QI-value combination* $x[QI]$ of an element $x \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ is the projection of x into the dimensions from $\mathcal{D}_{QI}$. The *CA-value combination* $x[CA]$ of an element x is defined analogously.
- The *sequences of QI-value combinations* inside a cluster C or over a database X are denoted as $C[QI]$ and $X[QI]$. The *sequences of CA-value combinations* are defined analogously.
- Concerning a database X, the *index set of confidential attributes* $X[CA]_x := \{i | x_i[CA] = x\}$ contains all indices of elements with CA-value combination x inside the sequence X.
- The *probability distributions of QI-values* inside a cluster C or over a database X are denoted as $P(C[QI])$ and $P(X[QI])$. The *probability distributions of CA-values* are defined analogously.

As we already discussed the problems and limitations of QI/CA separation, let us now just assume this classification of attributes is given and reasonable for a database X.

In 2007 Machanavajjhala et al. introduced the security notion ℓ-diversity [46]. Its goal is to prevent an attacker to get too much information about the confidential attributes of their target just by obtaining knowledge about the target's cluster membership. An attack scenario often referred to as *attribute disclosure attack*. The authors uncovered and addressed the following problem with k-anonymous data: Even if an attacker might not be able to distinguish between different elements of a cluster, they might obtain information about the confidential attributes of their target by comparing the distribution of confidential attribute values inside the target's cluster with the distribution on the whole database. In an extreme case, where a cluster is CA-uniform, an attacker might even obtain certainty about their target. Table 3.4 shows a database vulnerable to attribute disclosure attacks.

There are several variants of ℓ-diversity designed to mitigate the problem of attribute disclosure [46, 39]. The most basic definition is given as definition 3.10.

Definition 3.10 (Distinct ℓ-Diversity).
A k-anonymous database X categorized into quasi-identifiers QI and confidential attributes CA is called *ℓ-diverse* if in each cluster there are at least ℓ distinct CA-value combinations, i.e.

$$\exists x_1 \exists x_2 \ldots \exists x_\ell \left(\forall_{\{i,j\} \in \{1,\ldots,\ell\}} \left(i = j \vee x_i[CA] \neq x_j[CA] \right) \wedge \forall_i \left(X[CA]_{x_i[CA]} \geq 1 \right) \right).$$

Table 3.4: This 4-anonymous table highlights a problem called *attribute disclosure*. The frequency of the confidential attribute value 1 in the upper cluster is 1 while its frequency is 1/2 globally.

Element	QI-Attribute 1	QI-Attribute 2	CA-Attribute
x_1	7	4	1
x_2	7	4	1
x_3	7	4	1
x_4	7	4	1
x_5	3	7	2
x_6	3	7	2
x_7	3	7	3
x_8	3	7	4

Although distinct ℓ-diversity effectively prevents situations as depicted in table 3.4, a large majority of certain values inside a cluster is still allowed, and might still provide an attacker with some information about their target. An improved variant called *Entropy ℓ-diversity* provides greater protection at the cost of further limiting allowable clusters.

Definition 3.11 (Entropy ℓ-Diversity).
The *entropy of a cluster* C regarding its sequence of CA-value combinations $C[CA]$ is

$$E(C, CA) = -\sum_{p \in P(C[CA])} p \log p,$$

A k-anonymous database X is *entropy ℓ-diverse* if for every cluster C_j of X it holds $E(C_j, CA) \geq \log \ell$.

Entropy ℓ-diversity bars clusters from being allowed, just because they include some CA-outliers while being mostly homogeneous. However, if homogeneity of CA-values in a database X is high, entropy ℓ-diversity might be hard or even impossible to achieve. Further, as discussed in [39], neither form of ℓ-diversity is able to fully prevent attribute disclosure because it is still allowed to form clusters whose CA-distributions are very different from the CA-distribution over the whole database. Expressed in terms of information theory: The notion of ℓ-diversity might guarantee a high entropy of clusters, but does not give any guarantees about mutual information between the CA-distribution inside the cluster and globally. Hence, there is still a lot of possible information gain for an attacker, even on ℓ-diverse databases.

In an attempt to fix this, Li et al. introduced an anonymity notion called t-closeness [39]. This notion is the most restrictive among all syntactic anonymity notions. It only allows clusters whose distributions of confidential attribute values is close to the distribution of confidential attribute values on the whole database. The value t describes closeness between both distributions and is usually measured by Earth Mover's Distance.

Definition 3.12 (Earth Mover's Distance ([39])).
Let $P = (p_1, p_2, \ldots, p_m)$, $Q = (q_1, q_2, \ldots, q_m)$ and $d_{i,j}$ be the ground distance between

element i of P and element j of Q. Find a flow $F = [f_{i,j}]$ where $f_{i,j}$ is the flow of mass from element i of P to element j of Q that minimizes the overall work:

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{i,j} f_{i,j}$$

subject to the following constraints:

$$\begin{array}{ll} f_{i,j} \geq 0 & 1 \leq i, j \leq m \quad (c1) \\ p_i - \sum_{j=1}^{m} f_{i,j} + \sum_{j=1}^{m} f_{j,i} = q_i & 1 \leq i \leq m \quad (c2) \\ \sum_{i=1}^{m} \sum_{j=1}^{m} f_{i,j} = \sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1 & \quad (c3) \end{array}$$

These three constraints guarantee that P is transformed to Q by the mass flow F. The *Earth Mover's Distance (EMD)* is defined to be the total work of a minimal flow F, i.e.

$$EMD[P, Q] = \min_{F} WORK(P, Q, F).$$

EMD can be applied to numerical and categorical attributes as well. For numerical attributes, continuous values have to be approximated to obtain discrete probability distributions in C and X. Besides this inconvenience, ground distance between elements i and j is simply computed by the number of elements with probabilities between those for i and j. For categorical attributes, the ground distance between two elements could be e.g. the height of their closest common ancestor in the respecting hierarchy tree (as in the total information loss measure described above) or the size of their cca (like in the NCP measure).

Equipped with the EMD, we are now able to define t-closeness more formally:

Definition 3.13 (t-Closeness).
A cluster C within a database X is *t-close* if the distance between the distribution of a confidential attribute in C and the distribution of the attribute in X is no more than a threshold t, i.e.

$$EMD(P(C[CA]), P(X[CA])) \leq t.$$

A k-anonymous database X is *t-close* if each cluster $C \in \{C_1, \ldots, C_\ell\}$ is t-close, i.e.

$$\max_{1 \leq j \leq \ell} EMD(P(C_j[CA]), P(X[CA])) \leq t.$$

Of course, t-closeness could also be defined with other distance measures than EMD. However, as described by the authors in [39], there are some disadvantages when using e.g. total variation distance or Kullback-Leibler divergence. It should be further noted that t-closeness is generally considered to be private enough for most applications but not usable in practice, as its utility is low [11, 41]. Further, the authors of t-closeness initially did not specify an anonymization method to achieve this guarantee. However, in 2010 it was shown that the Mondrian algorithm [37] can be adapted to yield t-closeness [38]. Further, in 2015 it was shown that common microaggregation could be efficiently adapted to create t-close databases [65].

There are some interesting results regarding the connection between t-closeness and differential privacy. In 2013 Soria-Comas et al. showed that a limited form of ϵ-differential privacy using an uninformed attacker is achieved by an $\exp(\epsilon)$-close database, if instead of EMD, another distance measure closer related to semantic privacy is used [62]. In 2015 Domingo-Ferrer et al. showed that a probabilistic variant of t-closeness is achieved when differential privacy for CA-attributes is combined with k-anonymity for QI-attributes.

A slightly more utility-friendly variant of t-closeness called (n,t)-closeness was introduced in 2010 [38]. To avoid confusion with n as the number of elements in a database X, we will refer to it as (m,t)-closeness. The idea of (m,t)-closeness again is the anonymity concept of hiding in a crowd. Wheres k-anonymity describes hiding in a crowd of k individuals with identical quasi-identifiers, (m,t)-closeness further specifies hiding in a crowd of k individuals by requiring confidential attributes to appear as drawn from a similar distribution than that of at least m individuals in the database. Hence, $m \leq n$ where (n,t)-closeness is just t-closeness. Also n-anonymous databases are $(m,0)$-close for every m allowed.

Definition 3.14 ((m,t)-Closeness).
A cluster C within a database X is (m,t)-*close* if there exists a superset $\tilde{C} \supseteq C$ with $|\tilde{C}| \geq m$ and

$$EMD(P(C[CA]), P(\tilde{C}[CA])) \leq t.$$

A database X is (m,t)-*close* if it is k-anonymous for some value k, and each cluster $C \in \{C_1, \ldots, C_\ell\}$ of k or more elements with identical quasi-identifiers is (m,t)-close, i.e.

$$\max_{1 \leq j \leq \ell} EMD(P(C_j[CA]), P(\tilde{C}_j[CA])) \leq t.$$

Of course, the utility of t- or (m,t)-close databases could be measured with the same techniques as the utility of k-anonymous databases. Nevertheless, the authors introduced a novel approach to measure utility involving entropy. However, as visualized in table 3.5, closeness utility can be quite misleading in some cases.

Definition 3.15 (Closeness Utility).
Let X be a database anonymized into an (m,t)-close database $\hat{X}$ with clusters $C_1, \ldots, C_\ell$. Let $H(X)$ and $H(C_j)$ be the information entropy of the distribution of confidential attributes inside the database X and inside a cluster C_j, respectively.

The *closeness utility* of $\hat{X}$ is

$$U(\hat{X}) := H(X) - \sum_{1 \leq j \leq \ell} \frac{|C_j|}{|X|} H(C_j)$$

Safe k-Anonymization

After 2010 new semantic anonymity guarantees not reliant on QI/CA-separation were developed. In 2012 Li et al. introduced *safe k-anonymity* as a building block of a differentially private data publishing algorithm [40]. The main motivation of this approach is the dependence of clusters on the attribute values of their members. As extreme values could

Table 3.5: Three variants of 2-anonymous $(4, 0)$-close clusterings. It can be observed that perceived utility is not in accordance with closeness utility. Both, in database $\hat{X}_1$ and database $\hat{X}_2$ an observer obtains full information about the CA-distribution for both $QI = 1$ and $QI = 2$. However, closeness utility differs by the maximum possible amount. On the other hand, database $\hat{X}_3$ does provide significantly less information, but is evaluated like database $\hat{X}_1$.

Element	QI-Attribute	CA-Attribute
x_1	1	1
x_2	1	1
x_3	1	2
x_4	1	2
x_5	2	1
x_6	2	1
x_7	2	2
x_8	2	2

(a) Original database X; $H(X) = 1$

Element	QI	CA
x_1	1	1
x_2	1	1
x_3	1	2
x_4	1	2
x_5	2	1
x_6	2	1
x_7	2	2
x_8	2	2

(b) Database $\hat{X}_1$ with clustering $\mathcal{C}_1 = \{\{x_1, x_2\}, \{x_3, x_4\}, \{x_5, x_6\}, \{x_7, x_8\}\}$; $U(\hat{X}_1) = 1$

Element	QI	CA
x_1	1	1
x_2	1	1
x_3	1	2
x_4	1	2
x_5	2	1
x_6	2	1
x_7	2	2
x_8	2	2

(c) Database $\hat{X}_2$ with clustering $\mathcal{C}_2 = \{\{x_1, x_3\}, \{x_2, x_4\}, \{x_5, x_7\}, \{x_6, x_8\}\}$; $U(\hat{X}_2) = 0$

Element	QI	CA
x_1	1.5	1
x_2	1.5	1
x_3	1.5	2
x_4	1.5	2
x_5	1.5	1
x_6	1.5	1
x_7	1.5	2
x_8	1.5	2

(d) Database $\hat{X}_3$ with clustering $\mathcal{C}_3 = \{\{x_1, x_5\}, \{x_2, x_6\}, \{x_3, x_7\}, \{x_4, x_8\}\}$; $U(\hat{X}_3) = 1$

influence cluster statistics greatly, cluster membership of individuals with extreme values in one dimension might not be protected by the *hiding in a crowd* principle.

The remedy proposed for this problem requires the use of a data-independent clustering algorithm which is applied to all elements individually, without respecting the k-member clustering condition. To achieve k-anonymity anyway, a subsequent deletion of clusters with less than k elements is applied.

Definition 3.16 (Strongly-Safe k-Anonymization).
A *data-independent clustering* $\mathcal{C}^{ind} : \mathcal{D}_1 \times \cdots \times \mathcal{D}_d \to \{1, \ldots, \ell\}$ assigns a cluster to every possible database element x_i.
A *strongly-safe k-anonymization* is a three-step process.

1. A data-independent clustering $\mathcal{C}^{ind}$ is defined.
2. A database $\hat{X}$ is created out of a database X by applying $\mathcal{C}^{ind}$ and creating a database according to cluster statistics.
3. Clusters with less than k elements are removed from $\hat{X}$.

Remark 3.17.
Under the condition that cluster representatives are chosen as elements $x \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$, a strongly-safe k-anonymization is a size reducing anonymization.

As for ℓ-diversity and t-closeness, the perceived utility of the resulting databases is severely limited by the data-independent clustering. Not only might there be clusters with a lot more than k elements, which worsen cluster statistics, there might also be clusters with almost k elements, which have to be removed entirely.

Another downside of this approach is that it is unclear how to measure utility for safe anonymization algorithms. Depending on $\mathcal{C}^{ind}$ the same algorithm on the same database might deliver very different results. Obviously, information loss (see definition 2.23) cannot be applied, even on numerical data, as the removal of elements prevents the application of cost measures designed for SCA algorithms. C_{DM} or C_{AVG} might be applicable. However, it should be applied before data removal and clusters with less than k elements have to be sanctioned. As data removal might be anything from fine to detrimental depending on the use case, there seems to be no simple solution on how to evaluate element removal and hence utility of the resulting data.

Remark 3.18.
If instead of a data-independent clustering an ϵ-differentially private clustering is used, the resulting database might preserve more information and can still be used to achieve differential privacy on a database.

An anonymization scheme using ϵ-differentially private clustering is called *ϵ-safe k-anonymization*.

Insensitive Microaggregation

Insensitive Microaggregation is a concept introduced by Soria-Comas et al. in 2014 [64]. It describes a subset of common microaggregation algorithms that do not allow drastic changes in the clustering on similar databases. Clusters are allowed to change in only one element on databases that differ in at most one element. As a result the cluster centroids are more

stable and allow less information leakage about the membership of a particular element inside a cluster. Insensitive Microaggregation can be understood as a method creating databases with more privacy protecting than that offered by k-anonymity.

Definition 3.19 (Insensitive Microaggregation).
Let μ be a common microaggregation algorithm and $C_1^X, \ldots, C_\ell^X$ the (mutually disjoint) clusters μ creates when applied to a database $X \in \mathcal{X}_{n\times d}$.

Further let $C_1^{X'}, \ldots, C_\ell^{X'}$ be the clustering μ creates on a database $X' \in \mathcal{X}_{n\times d}$ which differs from X in only one element.

The algorithm μ is *insensitive* if for all choices of X and X' there is a bijection between the set of clusters $\mathcal{C}^X$ and $\mathcal{C}^{X'}$ such that each pair of corresponding clusters differs at most in a single element, i.e.

$$\forall X \forall X' \forall i \exists j : |C_i^X \cup C_j^{X'}| \leq \min\{|C_i^X| + 1, |C_j^{X'}| + 1\}.$$

As for safe k-anonymization, insensitive microaggregation is geared towards achieving differential privacy. Its main idea is to lower the sensitivity of data by applying insensitive microaggregation before using noise addition. As a lower sensitivity results in less noise needed, utility can be improved. However, as microaggregation itself introduces uncertainty as well, an improvement is not guaranteed and relies heavily on the implementation and data [64].

4

Complexity Results

To get a better understanding of the complexity of k-anonymous microaggregation within the context of other clustering problems, in the first section of this chapter we take a look at similarities and differences between microaggregation and the related problem of k-means clustering. Further, some well-known complexity results for k-means clustering are recited. As one of the main results of this thesis, in section 4.2 a reduction from planar 3-SAT to the decisional k-anonymous microaggregation problem shows NP-hardness of the microaggregation problem for all $k \geq 4$ and $d \geq 2$. Finally, in section 4.3 and section 4.4 additional complexity results and an overview of open problems in this area are stated.

4.1 The k-Means Clustering Problem

The k-anonymous microaggregation problem is similar to the well-known k-means clustering problem [47] traditionally tackled by the Lloyd algorithm [45]. Both problems define the search for a clustering with low distortion cost (see definition 2.13). However, in contrast to the k-anonymous clustering, for k-means a k-clustering instead of a k-member clustering is requested. Recall definition 2.9 for a definition of k-member clusterings and k-clusterings.

Definition 4.1 (Optimal k-Means Clustering Problem).
Given a database X, the *optimal k-means clustering problem* is to find a k-clustering $\mathcal{C}$ for X with minimum distortion cost.

Definition 4.2 (Decisional k-Means Clustering Problem).
Given a database X and a number $\gamma \in \mathbb{R}$, the *decisional k-means clustering problem* is to determine whether a k-clustering $\mathcal{C}$ with distortion cost at most γ exists for X.

An interesting property of k-means clustering is that optimal clusterings are guaranteed to produce Voronoi diagrams (see theorem 4.3). This characteristic proven in [10] and used for example in [34] is especially useful to limit search space when building algorithms to find optimal clusterings.

Theorem 4.3.
Let $\mathcal{C} = C_1, \ldots, C_\ell$ be an optimal k-means clustering for a database $X = (x_1, \ldots, x_n)$ and some integer k.

Every element x from X is at least as close to the centroid of its own cluster $C(x)$ as to any other cluster's centroid. More precisely,

$$\forall i \in \{1, \ldots, n\}\ \forall j \in \{1, \ldots, \ell\}\ \ \delta(x_i, c(C(x_i)))^2 \leq \delta(x_i, c(C_j))^2.$$

For k-anonymous microaggregation there is no similar result. As the example in figure 4.1 shows, there are scenarios in which the minimum cluster size constraint prevents elements from being clustered with close neighbors.

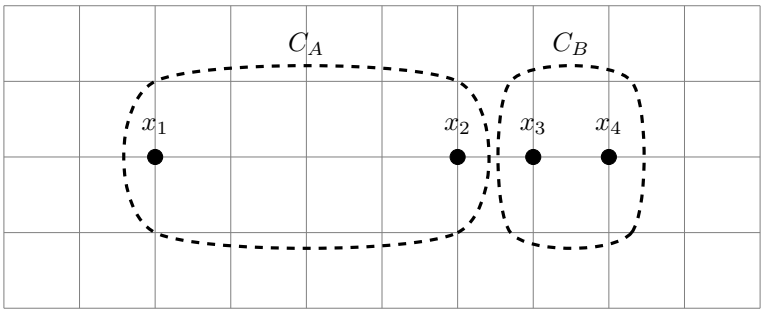


Figure 4.1: Visualization of an optimal 2-member clustering $\mathcal{C} = \{C_A, C_B\}$ for the database $X = (x_1, x_2, x_3, x_4) \in \mathcal{X}_{4\times 1}$. It can be observed that x_2 is closer to $c(C_B)$ than to $c(C_A)$ even though $x_2 \in C_A$. Hence, theorem 4.3 is not applicable for k-anonymous microaggregation.

Another major difference between the two clustering problems is the behavior towards multiset-respecting clusterings. Theorem 4.4, which is used but not proven in [48], states that any optimal k-means clustering is, or can easily be made, multiset-respecting. As no proof of theorem 4.4 seems to have been published so far, a proof is given below. As a direct consequence of the proof, corollary 4.5 limits the occurrences of non-multiset-respecting optimal k-means clusterings to rare and easily manageable edge-cases.

Theorem 4.4.

For any database X and any value k there is always an optimal k-means clustering $\mathcal{C}$ that is multiset-respecting.

Any optimal k-means clustering that is not multiset-respecting can be transformed into a multiset-respecting optimal k-means clustering in $O(n)$ time.

Proof. Let $\mathcal{C}$ be an optimal k-means clustering that is not multiset-respecting. There are two elements x_1, x_2 with $\delta(x_1, x_2)^2 = 0$ such that $x_1 \in C_A$ and $x_2 \in C_B$ for clusters $C_A \neq C_B$ in $\mathcal{C}$. Let $c_A := c(C_A)$ and $c_B := c(C_B)$ denote the centroids of both clusters. From theorem 4.3 we know that $\delta(x_2, c_A)^2 = \delta(x_2, c_B)^2 =: d^2$. In the following proof we use this fact to determine differences in cost when x_2 changes clusters from C_B to C_A. As our computation shows, the cost is reduced or stays the same independently of d^2 and $|C_A|$. Further we show that a cost equivalence is only possible for $|C_B| = 1$.

Let $C_{A+} := C_A \cup \{x_2\}$, $C_{B-} := C_B \setminus \{x_2\}$ be the new clusters and $c_{A+} := c(C_{A+})$, $c_{B-} := c(C_{B-})$ their new centroids. We further denote Euclidean distances $a := \delta(c_A, c_{A+})$, $b := \delta(x_2, c_{A+})$, $c := \delta(c_B, c_{B-})$ and their respective quadratic Euclidean distances a^2,b^2 and c^2. See figure 4.2 for an overview.

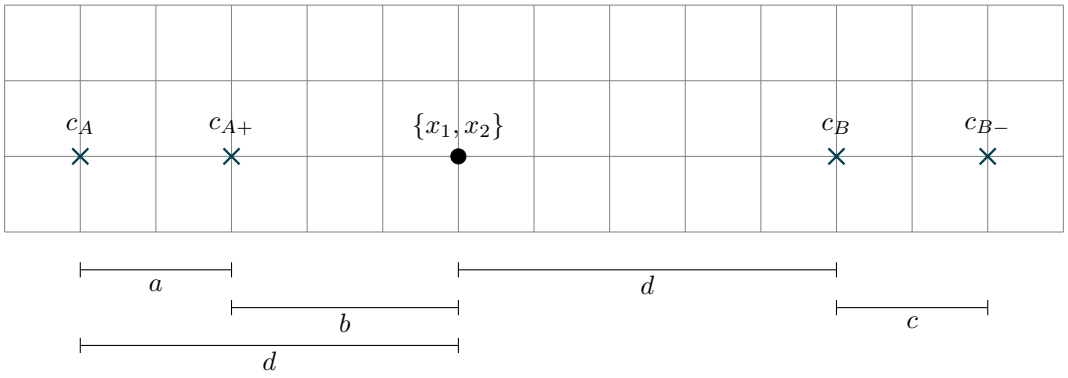


Figure 4.2: Relative positions of centroids c_A,c_{A+},c_B,c_{B-} and elements x_1,x_2. Relevant Euclidean distances are indicated by distance bars below. Other elements from C_A and C_B are not shown as their position is irrelevant for the proof. The distance d appears twice, as x_1 and x_2 lie exactly in the middle of the connecting line between c_A and c_B. Please note that all shown entities collapse to a single position if $|C_B| = 1$.

From corollary 2.20 we know that

$$\begin{aligned} cost(C_{A+}) &= cost(C_A) + |C_A| \cdot \delta(c_A, c_{A+})^2 + \delta(x_2, c_{A+})^2 \\ &= cost(C_A) + |C_A| \cdot a^2 + b^2 \end{aligned} \tag{4.1}$$

and

$$\begin{aligned} cost(C_{B-}) &= cost(C_B) - \left((|C_B| - 1) \cdot \delta(c_{B-}, c_B)^2 + \delta(x_2, c_B)^2\right) \\ &= cost(C_B) - \left((|C_B| - 1) \cdot c^2 + d^2\right). \end{aligned} \tag{4.2}$$

We consider two cases. For the first case assume $|C_B| = 1$. A simple computation using theorem 4.3 shows that the cost is not increased by the re-clustering of x_2.

$$\begin{aligned} & |C_A| \cdot a^2 + b^2 = (|C_B| - 1) \cdot c^2 + d^2 \\ \overset{|C_B|=1}{\Leftrightarrow}\ & |C_A| \cdot a^2 + b^2 = d^2 \\ \overset{Th.\ 4.3}{\Leftrightarrow}\ & |C_A| \cdot 0 + 0 \quad = 0. \end{aligned} \tag{4.3}$$

For the second case assume $|C_B| > 1$. Hence, to prove a cost reduction we need

$$|C_A| \cdot a^2 + b^2 < (|C_B| - 1) \cdot c^2 + d^2 \tag{4.4}$$

to hold. From corollary 2.22 we know that

$$c_{A+} = \frac{c_A \cdot |C_A| + x_2}{|C_A| + 1} \tag{4.5}$$

and (using $|C_B| > 1$)

$$c_{B-} = \frac{c_B \cdot |C_B| - x_2}{|C_B| - 1}. \tag{4.6}$$

Hence, we are able to express a and c in terms of d, $|C_A|$ and $|C_B|$ only.

$$a = \delta(c_A, c_{A+}) \overset{(4.5)}{=} \frac{||x_2 - c_A||_2}{|C_A| + 1} = \frac{d}{|C_A| + 1} \tag{4.7}$$

$$c = \delta(c_B, c_{B-}) \overset{(4.6)}{=} \frac{||x_2 - c_B||_2}{|C_B| - 1} = \frac{d}{|C_B| - 1} \tag{4.8}$$

To get rid of b^2 in (4.4), we use the relation

$$\begin{aligned} & b &&= d - a \\ \Leftrightarrow\ & b^2 &&= (d-a)^2 \\ \Leftrightarrow\ & b^2 &&= d^2 - 2ad + a^2 \\ \Leftrightarrow\ & b^2 &&= d^2 - \tfrac{2d^2}{|C_A|+1} + \left(\tfrac{d}{|C_A|+1}\right)^2. \end{aligned} \tag{4.9}$$

Using (4.7), (4.8) and (4.9) we obtain

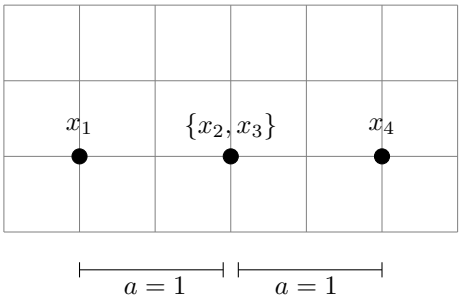
$$\begin{aligned} & (4.4) \\ \Leftrightarrow\ & |C_A| \cdot \left(\tfrac{d}{|C_A|+1}\right)^2 + d^2 - \tfrac{2d^2}{|C_A|+1} + \left(\tfrac{d}{|C_A|+1}\right)^2 &&< (|C_B| - 1) \cdot \left(\tfrac{d}{|C_B|-1}\right)^2 + d^2 \\ \Leftrightarrow\ & \tfrac{|C_A|d^2+d^2}{(|C_A|+1)^2} - \tfrac{2d^2}{|C_A|+1} &&< \tfrac{d^2}{|C_B|-1} \\ \Leftrightarrow\ & \tfrac{|C_A|+1}{(|C_A|+1)^2} - \tfrac{2}{|C_A|+1} &&< \tfrac{1}{|C_B|-1} \\ \Leftrightarrow\ & -\tfrac{1}{|C_A|+1} &&< \tfrac{1}{|C_B|-1}. \end{aligned} \tag{4.10}$$

It can be observed that re-clustering of x_2 from C_B into C_A cannot increase cost, and decreases cost whenever $|C_B| > 1$, i.e. when C_B has other elements beside x_2. As, by assumption, $\mathcal{C}$ is an optimal clustering, two elements on the same position can only be clustered into different clusters C and C' if $|C| = |C'| = 1$. A simple re-clustering algorithm taking an optimal clustering $\mathcal{C}$ and merging all elements at the same position in single clusters creates a multiset-respecting optimal clustering in $O(n)$ time. □

Note that empty clusters are created by the re-clustering operation used in the previous proof. Hence, if there were clusters C'' with $cost(C'') > 0$, the cost could be reduced by using an empty cluster to split C'', contradicting the optimality of the clustering. This further limits the occurrences of non-multiset-respecting optimal k-means clusterings to clusterings with a total cost of zero e.g. with clusters solely consisting of multiset elements from the same position.

Corollary 4.5.
An optimal k-means clustering $\mathcal{C}$ with $cost(\mathcal{C}) > 0$ is multiset-respecting.

Again, for k-anonymous microaggregation theorem 4.4 is not applicable. As the example in figure 4.3 shows, there are databases X and values of k for which there is no optimal clustering which is multiset-respecting.



$\mathcal{C}_j$	Cluster $C_1 \in \mathcal{C}_j$	Cluster $C_2 \in \mathcal{C}_j$	$\mathrm{cost}(\mathcal{C}_j)$
$\mathcal{C}_1$	$\{x_1, x_2\}$	$\{x_3, x_4\}$	$0.5+0.5 = 1$
$\mathcal{C}_2$	$\{x_1, x_4\}$	$\{x_2, x_3\}$	$2+0 = 2$
$\mathcal{C}_3$	$\{x_1, x_2, x_3, x_4\}$	$\{\}$	$2+0 = 2$

Figure 4.3: Visualization of a database $X \in \mathcal{X}_{4\times 1}$ for which an optimal 2-member clustering cannot be multiset-respecting. Clustering $\mathcal{C}_1$ with clusters $C_1 = \{x_1, x_2\}$ and $C_2 = \{x_3, x_4\}$ results in lower cost than both possible multiset-respecting clusterings $\mathcal{C}_2$ and $\mathcal{C}_3$. Note that this is not a contradiction to corollary 4.5, as a k-means clustering with $C_1 = \{x_1\}$ and $C_2 = \{x_2, x_3, x_4\}$ with a cost of $2/3$ is possible.

The differences discussed above seem to indicate that k-anonymous microaggregation is indeed a more difficult problem than k-means clustering. Considering complexity results for k-means clustering, we know the following: It has been shown in [48] that k-means clustering is NP-hard for $d = 2$ if k is variable. Further, from [2] we know that k-means clustering is NP-hard for $k = 2$ if d is variable. However, using theorem 4.3, Inaba et al. were able to prove that the k-means problem is solvable in polynomial time if both d and k are fixed [34].

It is easy to see that for k-means clustering as well as for k-anonymous microaggregation, NP-hardness for $d = 2$ dimensions implies NP-hardness for every $d > 2$ as any reduction function f creating two-dimensional databases X can be replaced by a reduction function f' using f and creating d-dimensional databases X' for $d \geq 2$ out of X where an element $x_i = (x_i^1, x_i^2)$ from X translates into an element $x_i' = (x_i^1, x_i^2, 0, \ldots, 0)$. As dimensions 3 to d from X' do not change the distortion cost of any clustering of X, f' is a valid reduction function in any reduction originally using f.

Analogously, for k-means clustering any reduction proving NP-hardness for $k = 2$ can be adapted to show NP-hardness for any $k \geq 2$. Consider a reduction function f creating a d-dimensional database X to be clustered into $k = 2$ clusters. An adapted reduction function f' takes X and adds additional elements in great distance to each other and to existing elements. For $k = 3$ one additional element is added, for $k = 4$, two, etc.

Considering k-anonymous microaggregation, hardness results for some values of k do not translate that easily into hardness results for arbitrary other values of k. On the one hand, this problem seems intuitively more difficult if the minimal size of a cluster is increased. On the other hand having to deal with less clusters may make things easier.

4.2 Microaggregation Is Hard for $k \geq 3$

In 2001 Oganian et al. claimed NP-hardness of the decisional k-anonymous microaggregation problem for fixed values of k and d [57]. However, a proof was given for the case $k = 3$

only. The authors used a reduction from *planar exact cover by 3-sets (planar X3C)* to show that there is a solution to a planar X3C-instance (Q, T) if and only if a 2-dimensional database X created by the reduction function $f(Q, T)$ can be clustered by a 3-member clustering with total cost equal to a fixed value computed from (Q, T). Further the authors give instructions on how to adapt the result for other values of d. By allowing arbitrary duplications of elements by the reduction function used, the result can be adapted to show NP-hardness for fixed values of k which are multiples of 3. Although this result indicates hardness for all fixed combinations of $d \geq 2$ and $k \geq 3$, no formal proof has been found so far.

During the work for this dissertation the missing proof has been found. In this section the reduction published in cooperation with Rüdiger Reischuk in [69] is presented. A preliminary version of this work has been published in [68].

The proof for theorem 4.6 is given in three parts. In the first part a reduction is used to show hardness of k-anonymous microaggregation for $k = 4$. Further, in the second and third part, the reduction function is adapted to show hardness for all even and all odd values of $k \geq 4$. Together with the original result of Oganian et al., theorem 4.6 is proven.

Theorem 4.6.
Decisional k-anonymous microaggregation is NP-hard for fixed values of $d \geq 2$ and $k \geq 3$.

The Case $k = 4$

The reduction is inspired by a reduction from planar 3-SAT to optimal planar k-means presented by Mahajan, Nimbhorkar and Varadarajan in 2009 [48]. A modified reduction function is used to take into account the differences between k-means clustering and k-anonymous microaggregation. The planar 3-SAT problem has been shown to be NP-hard by Lichtenstein in 1982 [43].

Definition 4.7 (Planar 3-SAT Problem).
Let F be a 3-CNF formula with variables $\{v_1, \ldots, v_{n'}\}$ and clauses $\{c_1, \ldots, c_m\}$. We call $G(F) := (V, E)$ the (undirected) graph of F, where

$$
\begin{aligned}
&V = \{v_i \mid 1 \leq i \leq n'\} \cup \{c_j \mid 1 \leq j \leq m\} \\
&E = E_1 \cup E_2 \text{ with} \\
&E_1 = \{(v_i, c_j) \mid v_i \in c_j \text{ or } \overline{v}_i \in c_j\} \\
&E_2 = \{(v_j, v_{j+1}) \mid 1 \leq j < n'\} \cup \{(v_{n'}, v_1)\}.
\end{aligned}
$$

If $G(F)$ is a planar graph, F is called a planar 3-CNF formula. The planar 3-SAT problem is to determine whether a given planar 3-CNF formula F is satisfiable.

The theorem is proven for the case of $k = 4$ by creating a database $X = X(F)$ with $d = 2$ attributes and a cost bound γ from a Planar 3-SAT instance F. Let F have n' variables v_i and m clauses c_j. For the correctness of this reduction we show that the 4-anonymous microaggregation instance $X(F)$ has a 4-member clustering $\mathcal{C}$ with $cost(\mathcal{C}) \leq \gamma$ if and only if F is satisfiable. This implies the claim for arbitrary $d \geq 2$ by fixing further attributes to constant values. Steps 1 to 6 show the construction steps of $X(F)$ performed by the reduction function f. Throughout the construction the planar 3-CNF formula $F = (\overline{v}_1 \vee v_2 \vee v_3) \wedge (v_2 \vee \overline{v}_3 \vee v_4) \wedge (v_1 \vee \overline{v}_2)$ is used as an example.

1. Consider $G(F)$ with all E_2 edges removed, i.e. $G'(F) = (V, E_1)$. Compute a planar embedding $\mathcal{E}$ of $G'(F)$ and for each variable vertex, assign labels $\kappa \in \{1, \ldots, m\}$ to incident edges according to a cyclic ordering. (See the left image of figure 4.4, edge labels are not shown.)
2. Replace every variable vertex v_i by a cycle Γ_i with m vertices $v_i^1, \ldots, v_i^m$. Reroute an edge e with label κ incident to v_i to v_i^κ. Observe that the embedding of $G'(F)$ can easily be translated into a planar embedding $\mathcal{E}'$ of the resulting graph $G''(F)$. (See the right image of figure 4.4.)

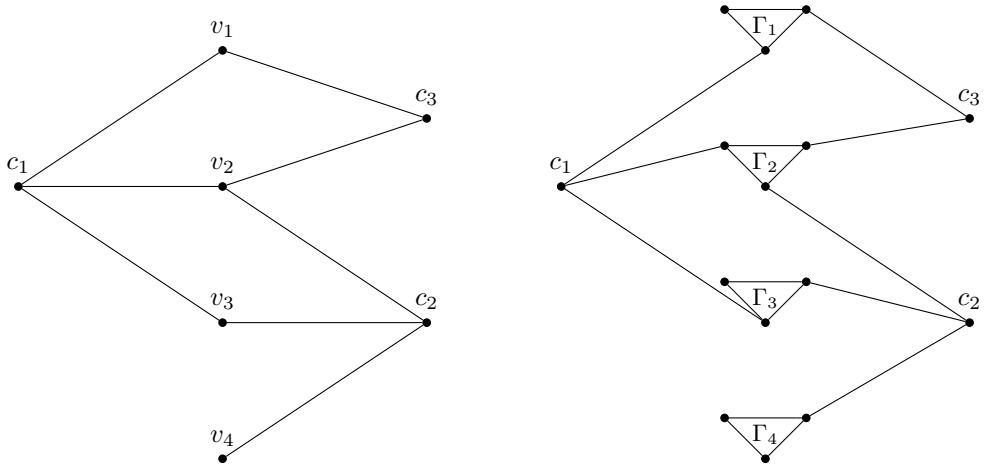


Figure 4.4: Steps 1 and 2 of the construction for $F = (\overline{v}_1 \vee v_2 \vee v_3) \wedge (v_2 \vee \overline{v}_3 \vee v_4) \wedge (v_1 \vee \overline{v}_2)$.

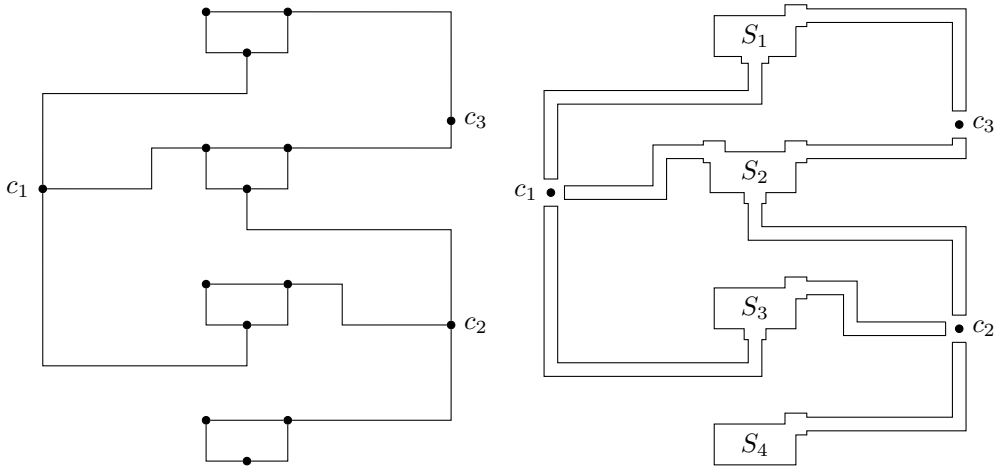


Figure 4.5: Overview of steps 3 and 4 of the construction.

3. Place every vertex of $G''(F)$ on the integer grid $\mathbb{Z}^2$ and route graph edges along edges of the grid ($\{(x, y), (x + 1, y)\}$ or $\{(x, y), (x, y + 1)\}$). Make sure that all vertices keep a minimum Euclidean distance from each other, so that all edges connecting clause vertices to a variable cycle are at least 4 grid squares apart. (See the left image of figure 4.5.) Later on, pairs of vectors that will be the elements of the database are derived from the grid points used for this embedding.

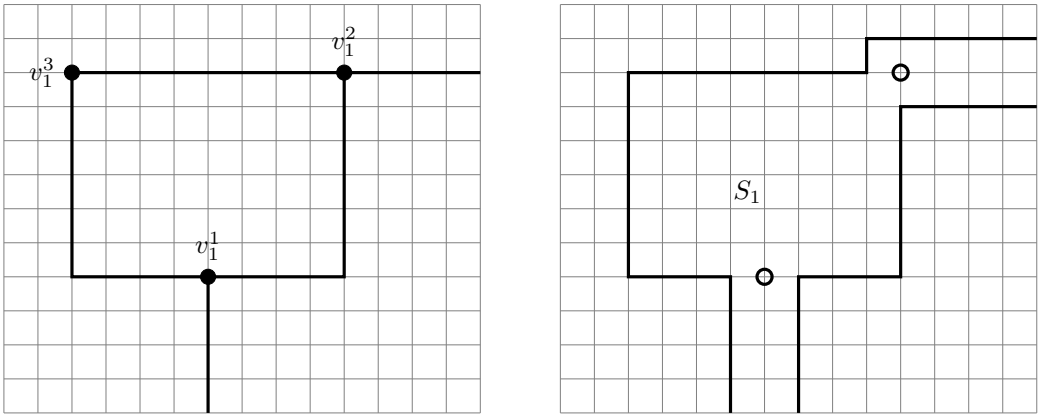


Figure 4.6: Step 4 of the construction around the previous position of vertex v_1. Note that non-adjacent points $p, p' \in P_i$ on a circuit S_i keep a minimum squared distance of at least $\delta(p, p')^2 = 2$ to each other.

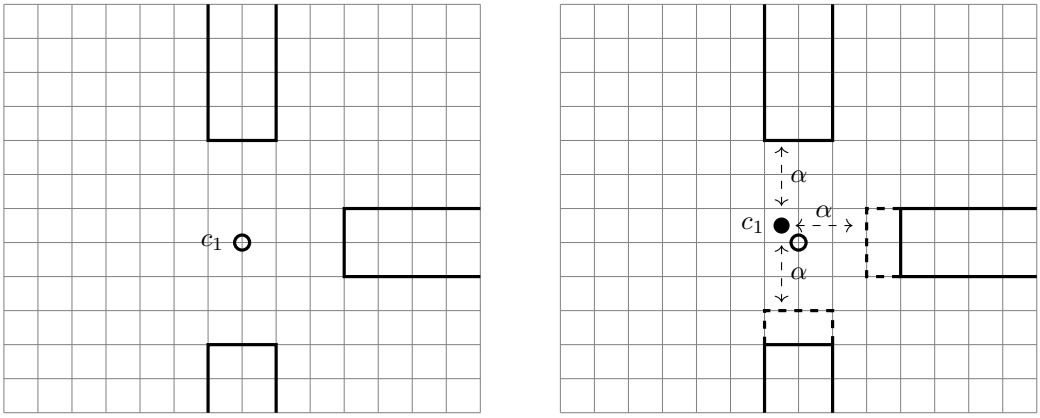


Figure 4.7: Step 4 of the construction around vertex c_1. After this step every clause vertex is in squared distance of α to each of its variable circuits. Note that any two points $p \in P_i$, $p' \in P_j$ of different circuits $S_i \neq S_j$ keep a minimum squared distance of at least $\delta(p, p')^2 = 2$ to each other.

4. Replace the embedding of an edge between a variable vertex v_i^κ and a clause vertex c_j by a pair of parallel rectilinear paths separated by two grid squares. On the variable vertex end, connect one path with the edge to $v_i^{\kappa-1}$ and the other path with the edge to $v_i^{\kappa+1}$, while keeping a minimum squared distance of 2 between any two points on separate paths. (See figure 4.6)

   At the end of the clause vertex, stop at distance 3 from c_j and connect both paths by a straight line to form a continuous circuit S_i for every variable. Move every clause vertex c_j (the white dot in figure 4.7) to the center of the north-west grid square touching it (the black dot). Extend all variable circuits S_i aiming at a clause vertex c_j so that the squared Euclidean distance between S_i and c_j is exactly $\alpha := (5/2)^2$. Obviously every circuit consists of an even number of grid edges. (See the right image of figure 4.5 for an overview and figure 4.6 and figure 4.7 for details at variable and clause vertices.)
5. For the next step the following definition is needed.

   Definition 4.8.
   Let P_i be the set of all grid points on a circuit S_i. Further, let $P = \bigcup_{i\in\{1,\dots,n'\}} P_i$. Two elements $p, p' \in P$ are *adjacent* iff both lie on the same circuit S_i and are adjacent in a circular ordering defined on P_i.

   Fact 4.9.
   The following properties are a result of construction steps 1–4.

   5.1 $|P_i|$ is even for every i.
   5.2 For every $p \neq p' \in P$ it holds $\delta(p, p')^2 \geq 1$.
   5.3 $\delta(p, p')^2 = 1$ iff p and p' are adjacent.
   5.4 $\delta(p, p')^2 \geq 2$ iff p and p' are not adjacent.

   There are two possible perfect matchings of the nodes $p \in P_i$ on any circuit S_i. Fix one of them and call it *true matching* from now on (the other one is subsequently called *false matching*). A clause vertex c_j belonging to a clause containing a variable v_i is either closer to a true-matching edge or a false-matching edge of the variable circuit S_i. If c_j contains v_i in positive form, and c_j is closer to a true matching edge of S_i, nothing needs to be changed. This holds analogously for negated variable and false matching edge. If, however, a matching is not according to the sign of a variable in a clause, the variable circuit is modified near the clause vertex, to correct this (see figure 4.8 and figure 4.9). The sets P_i are adapted in this case to include the 4 non-grid points that are now part of S_i, the results of fact 4.9 are, however, not affected.
6. To conclude the construction, database elements are created out of the resulting structure: To each $p \in P$ we assign $M = 2$ database elements $x_p, \tilde{x}_p$ whose attributes match the position of p.[1] To a clause vertex c_j a single database element x_{c_j} is assigned analogously. Let $\ell := \frac{|P|}{2}$ be half the number of circuit points. The cost threshold is set to $\gamma := \ell + \frac{4}{5} \cdot \alpha m$.

As stated in the construction, the squared distance between a circuit and a clause vertex is $\alpha = (5/2)^2$. The squared distance between a clause vertex and both endpoints of an edge

[1] We can avoid having identical database elements by separating x_p and $\tilde{x}_p$ by a tiny amount ϵ.

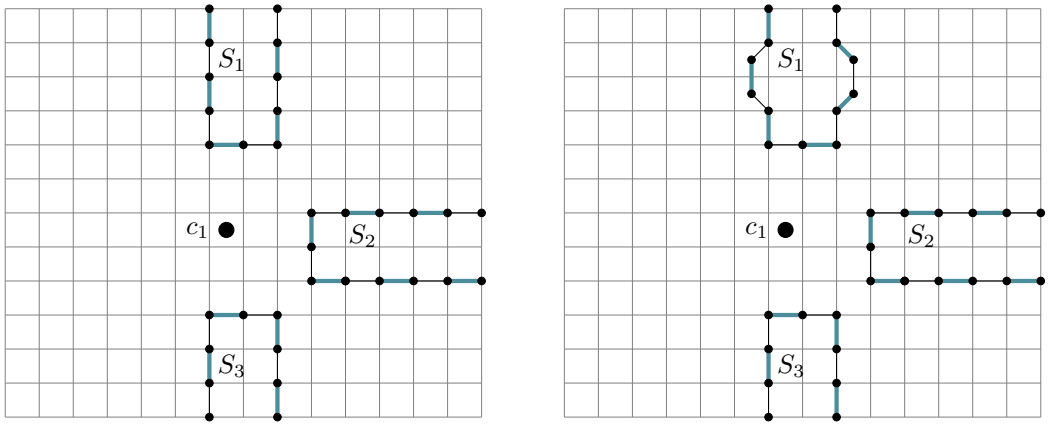


Figure 4.8: Step 5 of the construction around vertex c_1. Edges of true matchings are highlighted in blue. Note that the circuit S_1 has to be modified near c_1 because v_1 is in negative form in c_1 and c_1 is initially closer to a true matching edge. One the left: before the modification, on the right: after the modification.

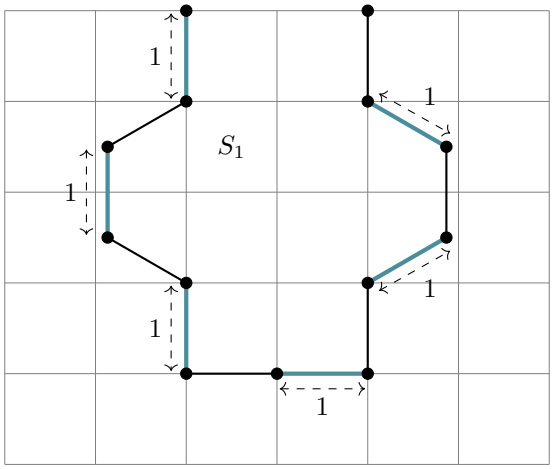


Figure 4.9: Step 5 of the construction around vertex c_1, zoom-in on the end of circuit S_1. Note that the matching on S_1 is unaffected in other regions and that the distances stated in fact 4.9 still hold.

of a circuit closest to it is $\alpha + 1/4$, all other circuit points are at least at squared distance $\alpha + 9/4$ from the clause vertex.

As there are $N = O(n' \cdot m)$ vertices in step 3 of the construction, a grid of side length N suffices. Thus, the number of database elements n is trivially bounded by $O(M \cdot (n')^2 \cdot m^2)$, where M denotes the number of duplicates corresponding to a grid point ($M = 2$ in the construction so far). This bound can be improved to $O(M \cdot n' \cdot m^2)$. The grid size can be further decreased by limiting the amount of variable vertices v_i^κ for a variable v_i by the actual degree of v_i in $G'(F)$. All elements of $\mathcal{X}(F)$ can be specified by binary strings of length bounded by $O(\log(n' \cdot m))$.

Lemma 4.10.
The formula F is satisfiable if and only if the associated database $X(F)$ has a 4-clustering of cost at most $\ell + \frac{4}{5}\alpha m$.

Proof of Lemma 4.10. ($\Rightarrow$) If the formula is satisfiable, every clause element can be clustered with pairs of circuit points nearby: Consider a satisfying assignment of F. If $v_i = 1$, cluster its circuit points according to the true matching and if $v_i = 0$ cluster according to the false matching. Because every clause c_j is satisfied, we are able to fix one of its satisfying variables v_i. Following the construction we are now able to cluster the clause element of c_j with two matched circuit points on the circuit for v_i which are at squared distance $\alpha + 1/4$ from the clause element of c_j. The cost of such a cluster C is

$$cost(C) = \frac{4 + 4\,(\alpha + 1/4)}{5} = 1 + \frac{4}{5}\alpha.$$

The total cost of the clustering is the sum of costs for m clusters of size 5 containing a clause element and the costs for $\ell - m$ clusters of size 4 containing both elements from two adjacent circuit points each. As the cost for each of these clusters is 1, the total cost is as claimed. See the left image of figure 4.10.

($\Leftarrow$) We now assume, F is not satisfiable. Hence a clustering as described above is not possible. To see that no clustering in this scenario can have cost smaller than $\ell + \frac{4}{5}\alpha m$, we consider possible alternative clusterings in this case.

1. The clustering is according to one of the perfect matchings on the circuits:
   At least one clause element c_j has to be included in a cluster from which at least one circuit point is not nearby i.e. in squared distance at least $\alpha + 9/4$ from c_j. See the right image of figure 4.10. The best clustering in this scenario has $m - 1$ clusters, each including two adjacent circuit points and a nearby clause element. As shown above, the cost of such a cluster is $1 + \frac{4}{5}\alpha$. Further there are $(\ell - m)$ standard clusters with cost of 1 each. One cluster remains. It has to incorporate four elements from two circuit points in squared distance 1 of each other and the clause element c_j in squared distances $\alpha + 1/4$ and (at least) $\alpha + 9/4$ from the circuit points. The cost of this cluster is

   $$\frac{4 + 2\,(\alpha + 1/4) + 2\,(\alpha + 9/4)}{5} = \frac{9}{5} + \frac{4}{5}\alpha.$$

   Altogether the total cost for such clustering is at least

   $$(\ell - m) + (m - 1)\left(1 + \frac{4}{5}\alpha\right) + \left(\frac{9}{5} + \frac{4}{5}\alpha\right) = \ell + \frac{4}{5}\alpha m + \frac{4}{5} = \gamma + \frac{4}{5}.$$

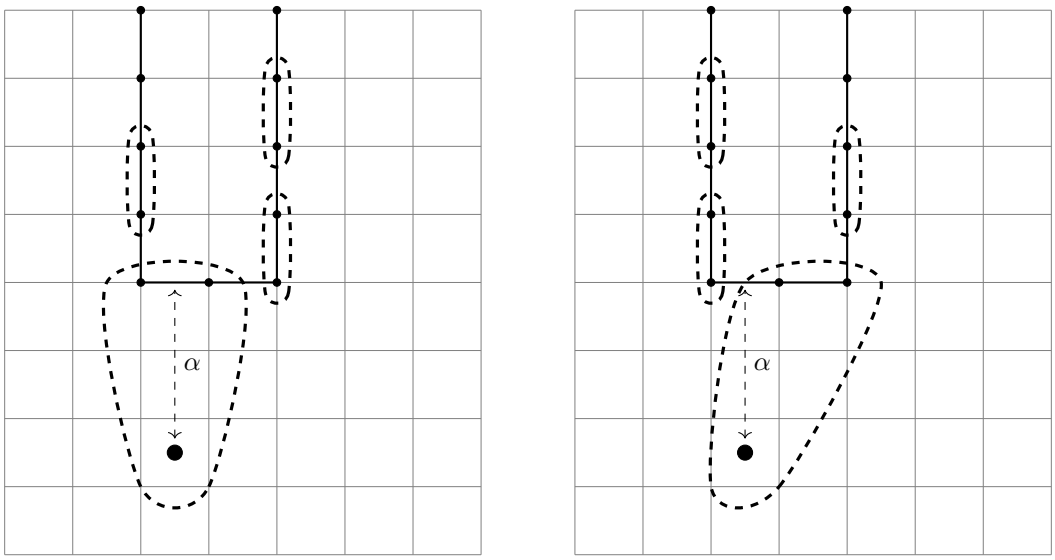


Figure 4.10: **On the left:** An optimal clustering for a satisfiable formula F clusters all clause vertices with the next closest pair of grid points (consisting of 4 elements altogether). **On the right:** If F is not satisfiable, a matching-based clustering has to incorporate at least one clause vertex in a cluster in greater distance.

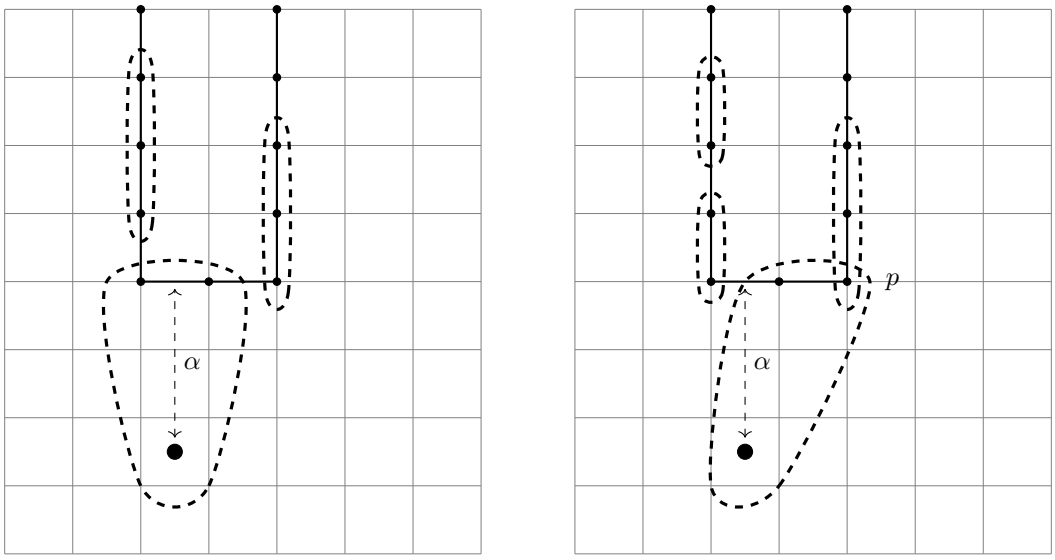


Figure 4.11: **On the left:** If F is not satisfiable but all clause vertices are clustered with two circuit points nearby, the clustering on that circuit cannot be matching-based. **On the right:** If F is not satisfiable, a clustering could violate the multiset-respecting property to decrease cost. In this case one of the two elements at a grid point p is clustered with a clause vertex, the other one is clustered with all elements of two adjacent circuit points.

2. The clustering on the circuits is not according to one of the perfect matchings. However, all clause elements are clustered with two circuit points nearby:

   This case is shown as the left image of figure 4.11. We obtain cost $m\left(1+\frac{4}{5}\alpha\right)$ for all clusters containing clause elements. To describe an optimal clustering for the remaining circuit points we have to make some assumptions.

   2.1 There is a variable assignment which satisfies all but one clause. (If this would not be the case, more clusters must differ from the clustering in ($\Rightarrow$), so the cost would rise.)

   2.2 Clusters consisting of elements from three circuit points can always be situated so that their outer elements are in squared distance of 2 to each other i.e. the cluster is around a corner of a circuit. (If this would not be the case, clusters containing elements from three circuit points would be more expensive.)

   The cheapest way to cluster the remaining elements is to cluster as many clusters as possible according to the matching belonging to the almost-satisfying variable assignment. We obtain a clustering which is according to the true-matching near clause elements which rely on the variable to be true and according to the false-matching near clause elements which rely on the variable to be false. For clause elements which are covered by other variable circuits we stick to the most convenient matching.

   Observation: The cheapest method to change between clusters which are according to one matching to clusters which are according to the other matching is by inserting a cluster containing all elements from three adjacent circuit points. Since we assumed that there is an assignment which satisfies all but one clause, we only need two of these clusters to meet our goals. The cost for a cluster containing all elements from three circuit points (using assumption b) is

$$\frac{4+4+8}{6}=\frac{8}{3}.$$

   Together with the clusters containing clause elements and the $(\ell-(m+2))$ matching clusters the total cost is at least

$$(\ell-(m+2))+m\left(1+\frac{4}{5}\alpha\right)+2\left(\frac{8}{3}\right)=\ell+\frac{4}{5}\alpha m+\frac{10}{3}=\gamma+\frac{10}{3}.$$

3. A clause element which cannot be clustered with nearby circuit points is clustered with both elements of a nearby circuit point and an additional element in squared distance $\alpha+9/4$. The other element from the split circuit point is incorporated in the adjacent matching-based cluster on the circuit, resulting in a cluster of 5 elements. The rest is according to the clustering in ($\Rightarrow$). See the right image of Figure 4.11.

   The cost for the cluster containing a clause point and 1.5 circuit points is

$$\frac{2+2(\alpha+\frac{1}{4})+(\alpha+\frac{9}{4})}{4}=\frac{3}{4}\alpha+\frac{19}{16}$$

   and the cost for the cluster of 2.5 circuit points is

$$\frac{4+2+4}{5}=2.$$

Hence, the total cost is

$$(\ell - (m+1)) + (m-1) \cdot (1 + \frac{4}{5}\alpha) + (\frac{3}{4}\alpha + \frac{19}{16}) + 2 = \ell + \frac{4}{5}\alpha m + \frac{19}{16} - \frac{1}{20}\alpha$$
$$= \ell + \frac{4}{5}\alpha m + \frac{7}{8} = \gamma + \frac{7}{8}.$$

□

Correctness of theorem 4.6 for $k = 4$ and $d = 2$ follows directly from Lemma 4.10. Hardness for $d > 2$ can simply be argued by introducing *dummy*-dimensions with only constant attribute values in any but two dimensions. Hence d can be increased without altering the complexity of the problem in any way.

Adaption to All Even $k \geq 4$

In this section the construction is adapted to obtain problem instances for larger even values of k. With this modification the correctness of theorem 4.6 for all $d \geq 2$ and all even $k \geq 4$ is shown. This modification changes two details from construction step 6: The number of elements on each grid point of the circuits is set to $M = k/2$ and the cost threshold to $\gamma := \ell \cdot \frac{k}{4} + \frac{k}{k+1}\alpha m$. This generalization results in an upper bound for the number of database elements of the form $n < O(k \cdot n' \cdot m^2)$. Following the same construction rules as above, for a satisfiable formula F the optimal cost is

$$(\ell - m) \cdot \frac{k}{4} + m \cdot \frac{(k/2)^2 + k \cdot (\alpha + 1/4)}{k+1}$$
$$= (\ell - m) \cdot \frac{k}{4} + m \cdot \left(\frac{k}{4} + \frac{k\alpha}{k+1}\right)$$
$$= \ell \cdot \frac{k}{4} + \frac{k}{k+1}\,\alpha m = \gamma.$$

To show correctness for a non-satisfiable formula F, cases 1, 2 and 3 of Lemma 4.10 need to be generalized as well.

1. In the case of a clustering according to one of the perfect matchings on the circuits, at least one clause element has to be clustered with $k/2$ elements in squared distance $(\alpha + 1/4)$ and $k/2$ elements in squared distance $(\alpha + 9/4)$. The resulting cost is at least

$$(\ell - m) \cdot \frac{k}{4} + (m-1) \cdot \left(\frac{k}{4} + \frac{k\alpha}{k+1}\right) + 1 \cdot \left(\frac{(k/2)^2 + k/2(\alpha + 1/4) + k/2(\alpha + 9/4)}{k+1}\right)$$
$$= (\ell - m) \cdot \frac{k}{4} + (m-1) \cdot \left(\frac{k}{4} + \frac{k\alpha}{k+1}\right) + \left(\frac{k}{4} + \frac{k(\alpha+1)}{k+1}\right)$$
$$= \ell \cdot \frac{k}{4} + \frac{k}{k+1}\,\alpha m + \frac{k}{k+1} = \gamma + \frac{k}{k+1}.$$

2. When all clause elements are clustered with two circuit points nearby, we need to change the clustering on the circuits. As for $k = 4$, the optimal clustering on the circuits is according to one of the perfect matchings, which is not achievable in this situation. Generalizing the setup from the $k = 4$ case we obtain cost

$$(\ell - (m+2)) \cdot \frac{k}{4} + m \cdot \left(\frac{k}{4} + \frac{k\alpha}{k+1}\right) + 2 \cdot \left(\frac{k^2}{\frac{3}{2}k}\right) = \ell \cdot \frac{k}{4} + \frac{k}{k+1}\,\alpha m + \frac{5}{6}k = \gamma + \frac{5}{6}k.$$

3. We now allow a clause element to be clustered with only $k-1$ other elements. The remaining element of a circuit point is clustered with all elements from the adjacent matching-based cluster. Otherwise the clustering is matching-based. The cost for this scenario is at least

$$(\ell - (m+1)) \cdot \frac{k}{4} + (m-1) \cdot \left(\frac{k}{4} + \frac{k\alpha}{k+1}\right) + \frac{(\frac{k}{2})^2 + \frac{3}{2}k}{k+1} + \frac{\frac{k}{2} \cdot (\frac{k}{2} - 1) + \frac{k}{2} \cdot (\alpha + \frac{1}{4}) + (\frac{k}{2} - 1) \cdot (\alpha + \frac{9}{4})}{k} = \frac{k^3\ell + k^2\ell + 4\alpha k^2 m + 8k^2 - 4\alpha - 6k - 9}{4k(k+1)}$$

which is greater than γ for $\frac{1}{8}(3 + \sqrt{281}) < 3 \leq k$.

As these three cases still represent the cheapest k-member clusterings when F is not satisfiable, hardness for all even $k \geq 4$ is proven.

Adaption to All Odd $k \geq 5$

The construction is further adaptable to fit odd values of $k \geq 5$. Instead of $M = k/2$ elements on each grid point of a circuit, now $\frac{k+1}{2}$ and $\frac{k-1}{2}$ elements are placed on adjacent grid points on circuits. This is possible because there is an even number of grid points on any circuit. The cost threshold is set to

$$\gamma = \frac{k^3\ell + k^2\ell + 4\alpha k^2 m - k\ell - \ell + m}{4k(k+1)}$$

which is precisely the cost of a matching based clustering for a satisfiable formula F:

$$(\ell - m) \cdot \frac{k^2 - 1}{4k} + m \cdot \frac{k^2 + 4\alpha k + k - 1}{4(k+1)} = \frac{k^3\ell + k^2\ell + 4\alpha k^2 m - k\ell - \ell + m}{4k(k+1)}$$

Again, we need to consider the cost of clusterings for unsatisfiable formulae F. As before in cases 1 and 2 the cost cannot be smaller than γ when clustering a clause vertex with a far-away circuit point or changing the matching based clustering on the circuits. Further, for our choice of α, non-multiset-respecting clusterings cannot undercut the cost of satisfiable clusterings either.

1. Cluster according to circuit matchings. At least one clause element causes increased clustering cost (see the right image of figure 4.10). The cost is at least

$$(\ell - m) \cdot \frac{k^2 - 1}{4k} + (m - 1) \cdot \frac{k^2 + 4\alpha k + k - 1}{4(k+1)} + \frac{k^2 + 4\alpha k + 5k - 5}{4(k+1)}$$

$$= \frac{k^3\ell + k^2\ell + 4\alpha k^2 m + 4k^2 - k\ell - 4k - \ell + m}{4k(k+1)}.$$

   This term is greater than γ for $k > 1$.
2. Cluster all clause elements with elements from two nearby circuit points (see the left image of figure 4.11). The cost is at least

$$(\ell - (m+2)) \cdot \frac{k^2 - 1}{4k} + m \cdot \frac{k^2 + 4\alpha k + k - 1}{4(k+1)} + 2 \cdot \frac{k(2k-2)}{3k-1}.$$

   This term is greater than γ for $3k^2 + 1 > 4k$, i.e. $k > 1$.
3. Cluster clause elements for non-satisfied clauses in a non-multiset-respecting way as depicted in the right image of figure 4.11. The cost is at least

$$(\ell - (m+1)) \cdot \frac{k^2 - 1}{4k} + (m - 1) \cdot \frac{k^2 + 4\alpha k + k - 1}{4(k+1)}$$

$$+ \frac{k^2 + 4\alpha k + 3k - 4\alpha - 16}{4k} + \frac{k^2 + 6k - 3}{4(k+1)}$$

$$= \frac{k^3\ell + k^2\ell + 4\alpha k^2 m + 8k^2 - 4\alpha - k\ell - 14k - \ell + m - 15}{4k(k+1)}.$$

   This term is greater than γ for $4\alpha + 14k + 15 < 8k^2$, i.e. for our choice of $\alpha = 6.25$ from the original construction: $k \geq 4$.

Summary

The construction and its adaptions cover all cases of fixed values of $k \geq 4$ and $d \geq 2$. Hence for a full proof of theorem 4.6 only the case $k = 3$ is missing. The reduction cannot easily be adapted to cover this case, as for $k = 3$ a non-multiset-respecting clustering can decrease cost of databases created from non-satisfiable formulae below that of satisfiable ones (see case 3 from the adaption to odd values of k). However, combining the result from Oganian et al. with the reduction and its adaptions shown in this section, theorem 4.6 is proven for all cases.

4.3 Other Complexity Results and Open Problems

In the previous section we discussed complexity of microaggregation for $d \geq 2$ and $k \leq 3$. In this section we take a look at the remaining cases as well as other complexity results regarding microaggregation.

$k = 2$

An obvious open question is the case $k = 2$. Our construction from section 4.2 cannot be extended to show hardness for $k = 2$. For $k = 2$ a cost reduction for the resulting database $X(F)$ is possible by clustering clause elements with only a single grid point, independently of the matching on circuits. A database created out of a non-satisfiable formula F therefore can be clustered with lower or equal cost compared to a matching based clustering for satisfiable F. For a strict 2-member clustering for which every cluster has to be of size exactly 2, the problem corresponds to finding a minimum weighted matching, which can be solved in time $O(n^{2.5})$ [55]. However, an optimal 2-member-clustering may also have clusters of size 3, thus this problem is more general than the matching problem.

$d = 1$

In 2003 Hansen and Mukherjee showed that optimal k-anonymous microaggregation can be solved in polynomial time for all fixed k and $d = 1$ [31]. Their proof is constructive and uses a shortest-path based algorithm finding an optimal solution in $O(\max(n \log n, nk^2))$.

A one-dimensional database X' is sorted in ascending order, resulting in a sorted database $X = (x_1, x_2, \ldots, x_n)$. Next, a weighted directed graph $G(X) = (V, E, w)$ is constructed. The set of nodes $V = \{0, 1, 2, \ldots, n\}$ contains the indices of elements in X as well as a node with label 0, acting as the source of paths in later steps. Edges exist between any two nodes (i, j) if x_j is between k and $2k$ elements after x_i in X, i.e. $E = \{(i, j) \mid i + k \leq j < i + 2k\}$.

A *corresponding cluster* $C_{(i,j)}$ for an edge (i, j) is defined as the cluster of elements $x_{i+1}, \ldots, x_j$. The weight of an edge (i, j) is defined as the cost of its corresponding cluster, i.e. $w(i, j) = \mathit{cost}(C_{(i,j)})$ for all $(i, j) \in E$. A path $P = (e_1, \ldots, e_m)$ on $G(X)$ is a sequence of edges $e_\ell \in E$ such that the second node of an edge $e_\ell = (v_a, v_b)$ is always the first node of the next edge $e_{\ell+1} = (v_b, v_c)$. No edge or node can occur twice on P. The length of a path P on $G(X)$ is defined as the sum of weights of its edges.

A *corresponding clustering* $\mathcal{C}(P)$ of a path P from node 0 to node n is the union of all clusters corresponding to edges on the path. An optimal solution for the k-anonymous microaggregation problem on X' is found by computing a shortest path in $G(X)$ from node 0 to node n. The shortest path P in $G(X)$ defines a clustering $\mathcal{C}(P)$ which is argued to be optimal. The following observations prove correctness of the algorithm:

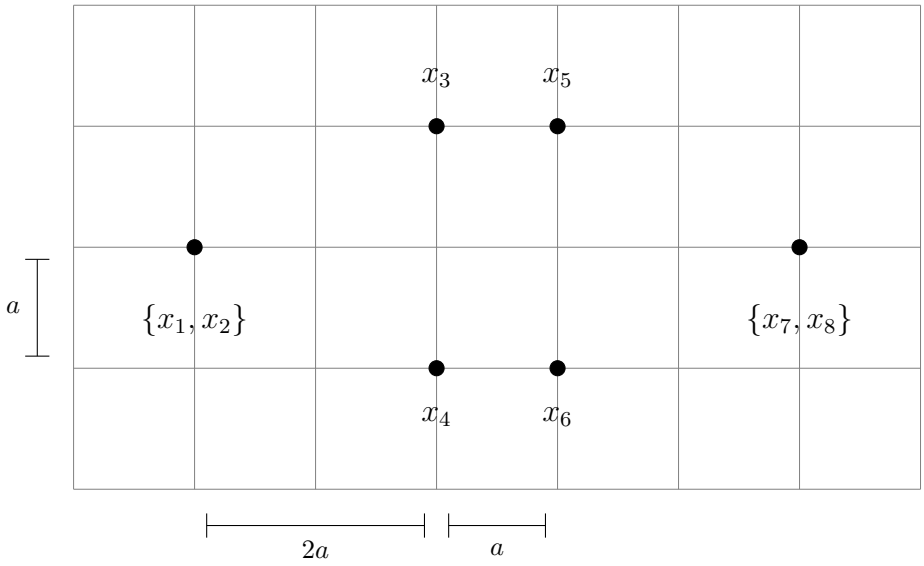
- Every cluster in an optimal clustering corresponds to an edge of $G(X)$.
- Every optimal clustering corresponds to a path from node 0 to node n in $G(X)$.
- The length of a path from node 0 to node n is the distortion cost of the corresponding clustering.

Sorting the database takes time $O(n \log n)$, graph construction takes time $O(nk^2)$ including weight computation and finding the shortest path takes time $O(nk)$ using the A^* algorithm. In total, the time complexity is $O(\max(n \log n, nk^2))$.

From k to k'

Another, more practical question is whether an optimal k-member clustering can be adapted to optimal $2k$-member or $k/2$-member clusterings by merging or splitting clusters, respec-

tively. As figure 4.12 shows, instances, for which this is not possible, exist. There are optimal 2-member clusterings which cannot be merged into optimal 4-member clusterings and analogously there are 4-member clusterings which cannot be split into optimal 2-member clusterings.



$\mathcal{C}_j$	Cluster $C_1 \in \mathcal{C}_j$	Cluster $C_2 \in \mathcal{C}_j$	Cluster $C_3 \in \mathcal{C}_j$	Cluster $C_4 \in \mathcal{C}_j$	$\text{cost}(\mathcal{C}_j)$
$\mathcal{C}_1$	$\{x_1, x_2\}$	$\{x_3, x_5\}$	$\{x_4, x_6\}$	$\{x_7, x_8\}$	$0 + {a^2}/{2} + {a^2}/{2} + 0 = a^2$
$\mathcal{C}_2$	$\{x_1, x_2, x_3, x_4\}$	$\{x_5, x_6, x_7, x_8\}$	$\{\}$	$\{\}$	$2 \cdot {24a^2}/{4} = 12a^2$
$\mathcal{C}_3$	$\{x_1, x_2, x_3, x_5\}$	$\{x_4, x_6, x_7, x_8\}$	$\{\}$	$\{\}$	$2 \cdot {31a^2}/{4} = 15.5a^2$
$\mathcal{C}_4$	$\{x_1, x_2, x_7, x_8\}$	$\{x_3, x_4, x_5, x_6\}$	$\{\}$	$\{\}$	${100a^2}/{4} + {20a^2}/{4} = 30a^2$

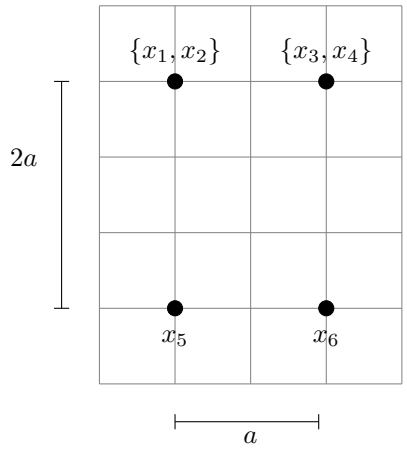
Figure 4.12: Visualization of a database $X \in \mathcal{X}_{8\times 2}$ for which $\mathcal{C}_1$ is an optimal 2-member clustering and $\mathcal{C}_2$ is an optimal 4-member clustering. $\mathcal{C}_3$ and $\mathcal{C}_4$ are the only (somewhat reasonable) 4-member clusterings that can be created by merging existing clusters from $\mathcal{C}_1$. However, it can be observed that neither $\mathcal{C}_3$ nor $\mathcal{C}_4$ can achieve optimal cost. Further, by splitting clusters from $\mathcal{C}_2$, no optimal 2-member clustering can be created. By adjusting the scaling factor a, additional cost needed for applying the merging / splitting approach can be shown to have no upper bound.

$$k = \lfloor {n}/{2} \rfloor$$

For the case $k = \lfloor \frac{n}{2} \rfloor$ one might ask, whether an optimal 2-means clustering can be adapted to obtain optimal k-member clusterings. As figure 4.13 shows, an optimal 2-means clustering cannot always be adapted to obtain an optimal $\lfloor {n}/{2} \rfloor$-member clustering just by transferring elements from the bigger to the smaller cluster. Instead, both clusters might need to swap elements, which nullifies any potential advantage of a k-means clustering as a preprocessing step of a k-anonymous microaggregation algorithm guaranteeing optimality.

Another idea might be to divide a database by a hyperplane through its geometrical representation, as the problem would then be reduced to the problem of finding such optimal

hyperplane. However, as figure 4.3 shows, convex hulls of optimal clusters might not be disjoint. i.e. a hyperplane might not be sufficient to separate two optimal clusters.



$\mathcal{C}_j$	Cluster $C_1 \in \mathcal{C}_j$	Cluster $C_2 \in \mathcal{C}_j$	cost$(\mathcal{C}_j)$
$\mathcal{C}_1$	$\{x_1, x_2, x_3, x_4\}$	$\{x_5, x_6\}$	$4a^2/4 + a^2/2 = 1.5a^2$
$\mathcal{C}_2$	$\{x_1, x_2, x_5\}$	$\{x_3, x_4, x_6\}$	$2 \cdot 8a^2/3 = 16a^2/3$

Figure 4.13: Visualization of a database $X \in \mathcal{X}_{6\times 2}$ for which $\mathcal{C}_1$ is an optimal 2-means clustering and $\mathcal{C}_2$ is an optimal 3-member clustering. It can be observed, that $\mathcal{C}_1$ cannot be transformed into an optimal 3-member clustering by moving elements from a bigger to a smaller cluster.

4.4 Related Results

The high complexity of k-anonymous clustering is not limited to microaggregation algorithms using distortion cost as optimization goal. In fact, for a number of related problems, hardness results are known. In this section we take a closer look at some of the most cited complexity results stating hardness of achieving k-anonymity with respect to several optimization goals and anonymization strategies.

Complexity of Suppression and Generalization

In 2004, Meyerson and Williams analyzed the complexity of k-anonymous suppressors on databases potentially including non-numerical or duplicate elements [54]. Databases and k-anonymity are defined analogously to definition 3.3 and definition 3.4. A suppressor is allowed to replace any element attributes by empty strings ε. A suppressor does not rely on clustering and is hence conceptually different to a local recoding scheme. In fact, a suppressor achieving k-anonymity might not be representable by a k-member clustering algorithm, as suppression happens on an individual basis. The authors prove NP-hardness for two variants of suppressors and the utility goal of suppressing as few attribute values as possible.

Definition 4.11 (Decisional k-Anonymous Suppressor Problem).
Given a database $X \in \mathcal{D}_{n\times d}$ and a number $\gamma \in \mathbb{N}$, the *decisional k-anonymous suppressor*

problem is to determine whether X can be made k-anonymous by suppressing at most γ attribute values.

Theorem 4.12 ([54]).
The decisional k-anonymous suppressor problem is NP-hard for fixed $k \geq 3$ and variable d.

Proof sketch. The k-dimensional perfect matching problem[2] is reduced to the decisional k-anonymous suppressor problem. Given a k-uniform hypergraph $H = (V, E)$ with $|V| = n$ and $|E| = d$, a database $X \in \{0, 1\}_{n \times d}$ of binary attributes is created. Every element x_{v_i} of X corresponds to a node $v_i \in V$ and every attribute dimension corresponds to an edge $e_j \in E$. When $v_i \in e_j$, $x_{v_i}^{e_j} = 0$, otherwise $x_{v_i}^{e_j} = 1$. When a perfect matching $M \subseteq E$ in H exists, a suppressor is able to achieve k-anonymity by suppressing every attribute value except for those corresponding to a node's matching edge. As a result, $n(d-1)$ attribute values are suppressed and the result is k-anonymous due to the k-uniformity of H. However, if no perfect matching exists, more suppressions are needed to achieve k-anonymity. Hence, by deciding whether a database can be made k-anonymous by suppressing at most $n(d-1)$ attribute values, we are able to decide, whether a perfect matching on H exists and hence are able to solve the k-dimensional perfect matching problem. □

Definition 4.13 (Decisional k-Anonymous Attribute Suppressor Problem).
Given a database $X \in \mathcal{D}_{n \times d}$ and a number $\gamma \in \mathbb{N}$, the *decisional k-anonymous attribute suppressor problem* is to determine whether X can be made k-anonymous by suppressing at most γ attribute dimensions in every element of X.

Theorem 4.14 ([54]).
The decisional k-anonymous attribute suppressor problem is NP-hard for fixed $k \geq 3$ and variable d.

Proof sketch. The proof is similar to that of theorem 4.12. Given a k-uniform hypergraph $H = (V, E)$, a database X is created in the same way as before. As attributes correspond to edges, the suppression of an attribute j in every element results in a database corresponding to a hypergraph $H' = (V, E')$ for which $E' = E \setminus \{e_j\}$. When H has a perfect matching, the removal of $d - n/k$ attributes corresponding to the non-matchings edges results in a k-anonymous database. However, if H does not have a perfect matching, more attributes need to be suppressed. □

Suppressing attributes or even whole dimensions of databases is a valid approach to achieve k-anonymity. However, as discussed in section 3.2 allowing generalization according to hierarchy trees offers a more fine-grained approach to suppress information. Further, using the NCP or Total-IL allows a more detailed image of information loss, applying generalization and suppression. Nevertheless, according to [75] and [12] as NCP and Total-IL optimizing generalizations are generalized approaches of minimal suppression, these more sophisticated approaches are NP-hard, as well.

[2] Equivalent to exact cover by k-sets, NP-hard for $k \geq 3$ [29]

Complexity of Cluster Size

In 2006, LeFevre et al. introduced the recoding scheme *Mondrian* producing k-anonymous clusterings with good quality according to C_{DM} and C_{AVG} [37]. Mondrian is able to operate as a global or local recoding scheme. By using a reduction from the partition problem, NP-compleness of determining the existence of global recoding schemes given a fixed C_{AVG} bound is proven. As stated by the authors, this result is adaptable to C_{DM} as well.

Definition 4.15 (Decisional k-Anonymous Cluster Size Problem).
Given a database $X \in \mathcal{D}_{n \times d}$ and a number $\gamma \in \mathbb{N}$, the *decisional k-anonymous cluster size problem* is to determine whether X can be made k-anonymous by using a global recoding scheme with a normalized average equivalence class size $C_{AVG} \leq \gamma$.

Theorem 4.16 ([37]).
The decisional k-anonymous cluster size problem is NP-complete for variable d and k.

Proof sketch. Given an instance of the partition problem, i.e. a set A of d positive integers $\{a_1, \ldots, a_d\}$, create a database X with elements $x \in \{0,1\}^d$ such that for every element $a_i \in A$ there exist a_i database elements $x = (x_1, \ldots, x^{i-1}, x^i, x^{i+1}, \ldots, x^d) = (0, \ldots, 0, 1, 0, \ldots, 0)$ consisting of a 1 in dimension i and 0s in every other dimension. Hence, $X \in \{0,1\}_{\sum_{i=1}^{n} a_i \times n}$. Further, let $k = \frac{\sum_{i=1}^{n} a_i}{2}$ and $\gamma = 1$.

When A is a positive instance of partition, there exists a set $A' \subseteq A$ such that $\sum_{a_i \in A'} a_i = \sum_{a_j \in A \setminus A'} a_j$. Clustering all database elements x created from elements $a \in A'$ in a cluster C_1 and all other database elements in a cluster C_2 results in a multiset-respecting k-member clustering with cluster sizes $|C_1| = |C_2| = k$ and hence $C_{AVG} = 1$. If, however, A is a negative partition instance, any multiset-respecting k-member clustering of X must result in a single cluster containing all database elements, so $C_{AVG} = 2$. This concludes the reduction and implies NP-hardness.

For membership in NP simply observe polynomial-time verifiability of a solution candidate to the decisional k-anonymous cluster size problem. □

Of course, the hardness result is only applicable for global recoding schemes, as the reduction requires a restriction to multiset-respecting clusterings. Deciding whether a local recoding scheme exists with a given C_{AVG} or C_{DM} bound is trivial, as no restriction on which elements can or cannot be clustered together is made. Hence, hardness originates from a limitation to multiset-respectiveness. In comparison, microaggregation is NP-hard regardless of such restriction, as discussed above.

5

Maximum Distance Heuristics

Univariate k-anonymous microaggregation can be solved efficiently in $O(\max(n \log n, nk^2))$ time as shown in section 4.3. However, multidimensional k-anonymous microaggregation is NP-hard for $k \geq 3$ and $d \geq 2$. Early microaggregation heuristics used univariate approaches to solve the microaggregation problem for higher dimensional data as well. By assuming the dimensions are statistically independent, d instances of univariate microaggregation approaches can anonymize d-dimensional databases (see e.g. [14]). As a result one obtains d univariate, k-anonymous clusterings. Obviously, this approach has its limits, as in most cases multidimensional data is compiled *because* of potential dependencies between attribute dimensions. Hence, utility of the anonymized data for statistical analysis is severely impaired. Another primitive method proposed in [51] is to project multidimensional data onto a single dimension by using principal component analysis or other projection techniques. While this procedure might preserve some global dependencies between attributes, diverse local clusters with few elements can hardly be preserved by such representation.

In this chapter we take a closer look at a class of multivariate microaggregation heuristics called *maximum distance heuristics*. Before going into more detail, some notations need to be introduced. For the clustering approaches introduced in this chapter, the order of elements inside a database X or a cluster C is irrelevant. To simplify notations we consider databases and clusters as sets of unique elements instead of sequences and index sets. The usual set operations $(\in, \subseteq, \setminus, \cup, \cap)$ are used to indicate inclusion or exclusion of elements in clusters. Keep in mind that databases including several identical-valued elements are still allowed and all properties of clusters defined in chapter 2 are still valid, as the simplified notation can easily be adapted to match the original one defined e.g. in definitions 2.1 and 2.9.

Definition 5.1 (Neighborhood and Closest Clusters).
Let X be a database and $\mathcal{C}$ a (potentially incomplete) clustering of X.

- The *nearest neighbor* of an element $x \in X$ according to $\delta(\cdot,\cdot)^2$ within a set of elements S is noted $\nu(x, S)$.
- The *ℓ-neighborhood* of an element $x \in X$ within a set of elements S, noted $N_\ell(x, S)$ is a cluster of size $\ell + 1$ including x and its ℓ nearest-neighbors within S according to $\delta(\cdot,\cdot)^2$.
- An *unassigned element* is an element $x \in X$ which has not jet been assigned to a cluster $C \in \mathcal{C}$ during a clustering process. The *set of unassigned elements* U contains all such elements.

– The *closest cluster* of an element $x \in X$ is the cluster

$$clos(x) := \arg\min_{C \in \mathcal{C}} \delta(x, c(C))^2.$$

In 1998 Mateo-Sanz and Domingo-Ferrer introduced the first truly multivariate microaggregation heuristic called MD (Maximum Distance) [51, 18]. Instead of dividing the problem into independent subproblems or projecting a d-dimensional database onto a single dimension, multidimensional data is handled in all dimensions simultaneously.

The algorithmic idea of MD is quite simple: When there are at least $2k$ elements, find the two elements x_r and x_s in greatest distance to each other and build clusters C_r and C_s out of x_r and x_s by clustering them with their respective $k-1$ nearest neighbors. Repeat until there are less than $2k$ elements. If between k and $2k-1$ elements are remaining, create an additional cluster with all remaining elements, otherwise assign all remaining elements x to their closest existing cluster $clos(x)$. How $clos(x)$ is computed exactly is not disclosed by the authors of MD. A reasonable choice from a complexity point of view would be to look for the minimum distance between the element and the cluster centroids, as defined in definition 5.1. For an analysis of alternative methods of computing $clos(x)$ see section 5.1. For a pseudocode of MD see algorithm 1.

Algorithm 1: MD [51]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

```
Let U ← X
Let C ← ∅
repeat
    Let x_r, x_s ∈ U be the unassigned elements in greatest distance to each other
    C ← C ∪ {N_{k−1}(x_r, U)}
    U ← U \ N_{k−1}(x_r, U)
    C ← C ∪ {N_{k−1}(x_s, U)}
    U ← U \ N_{k−1}(x_s, U)
until |U| < 2k
if |U| ≥ k then
    C ← C ∪ {U}
else
    foreach x ∈ U do
        clos(x) ← clos(x) ∪ {x}
```

Any cluster in the resulting clustering has between k and $2k-1$ elements. Further, for large databases most clusters would be of size exactly k, as the algorithm performs a strict k-member clustering up until the last $2k-1$ elements are handled. However, as several elements might be added to their closest clusters in the last step, the result might contain several clusters of more than k elements, which prevents the algorithm from being categorized as fixed-size. As the original paper lacks a detailed time complexity analysis, the complexity of MD is evaluated in the proof of theorem 5.2.

Theorem 5.2.
The worst case time complexity of algorithm 1 is $O(n^2 \cdot \max\{n/k, d\})$.

Proof. As distance computations between two elements take $O(d)$ and the distances between any two elements need to be computed, the computational time complexity of MD is at least $O(n^2 d)$. Additional time is needed, because step 4 to 8 are executed $O(n/k)$ times. A naive implementation of step 4 would sort all distances in the beginning and remove distances between assigned elements from the sorted list each time clusters are formed. Sorting takes time $O(n^2 \log n^2)$. However, removing assigned elements from the sorted list takes time $O(n^2)$ each time clusters are formed. This results in an additional time complexity of $O(n^2 \log n^2) + O(n^3/k)$. Without sorting, a linear search through the distance matrix is needed in each execution of step 4, which results in a time complexity of $O(n^3/k)$ and shows that sorting is not beneficial. Hence, a total time complexity of MD of $O(n^2 \cdot \max\{n/k, d\})$ can be achieved. As for all common microaggregation algorithms, $O(nd)$ is needed to compute cluster centroids and create an anonymized database out of a k-member clustering. As this process is the same for all microaggregation algorithms and additional time complexity is majorized by the clustering parts of all currently known multidimensional microaggregation heuristics, this part of the time complexity of microaggregation heuristics is omitted in time complexity analysis from now on. Space complexity of MD is bounded by $O(n^2)$ as a matrix of distances needs to be stored in order to avoid recalculation of distances. □

As MD and other heuristics presented in this chapter are not known to achieve any approximation guarantees, their performance is measured in terms of information loss and runtime on several real and synthetic benchmark databases. In accordance with fact 2.24 information loss is reported in percentages, where 0% means no information loss and 100% a worst case information loss, equal to that of an n-member clustering. See section 5.2 for detailed information on hardware, software, databases and parameters used to conduct the experiments presented in this chapter. Results of MD compared to other microaggregation heuristics are presented in tables 5.3 to 5.6 as well as in appendix A.

5.1 Maximum Distance to Average Vector

Over the years, MD received several incremental updates, resulting in the algorithms MDAV (Maximum Distance to Average Vector) [20], MDAV$^+$ [61, 70], V-MDAV [61], and MDAV* [70] which achieved competitive information losses and time complexities compared to other microaggregation algorithms of their time. As all these algorithms are based on MD and use distance computations to find cluster-generating elements, we refer to them as the class of *maximum distance heuristics* from now on.

MDAV

A core idea behind MD is to create clusters in greatest distance to each other. This technique is inspired by the one-dimensional clustering algorithm k-ward ([51]) which performs well, because outliers are handled first. As outliers need to be included in some clusters, it might

be better to include them right away to avoid a large sparely populated (and hard to cluster) domain of elements later on. However, selecting the two elements with maximum distance between each other is computationally expensive and ineffective when looking for outliers in more than one dimension.

MDAV is an evolutionary improvement to MD reducing time complexity by using a different strategy to find outliers. As the name suggests, in each round MDAV calculates the centroid $c(U)$ of all remaining unassigned elements $x \in U$ and selects the element x_r in greatest distance to $c(U)$. As for MD, MDAV chooses x_s as the element in greatest distance to x_r and clusters x_r as well as x_s with their $k-1$ nearest neighbors. Another difference between MD and MDAV is the handling of the last elements. MDAV stops its initial behavior when less than $3k$ elements remain unassigned. If there are between $3k-1$ and $2k$ elements left, a global centroid c and most distant element x_r are computed a last time. Now, two clusters are created: One including x_r and its $k-1$ nearest neighbors and another one with all other elements remaining. However, if there are less than $2k$ elements left after the main loop of MDAV, all remaining elements are clustered together in a single new cluster. In contrast to MD, the resulting clustering is fixed-size, as only one cluster might have more than k elements. Anonymity of the result is guaranteed, as after the main loop not less than k elements remain. A pseudocode of MDAV is given in algorithm 2.

Algorithm 2: MDAV [20]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
repeat
  Let $c(U)$ be the centroid of U
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(U))^2$
  Let $x_s \leftarrow \arg\max_{x \in U} \delta(x, x_r)^2$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_r, U)\}$
  $U \leftarrow U \setminus N_{k-1}(x_r, U)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_s, U)\}$
  $U \leftarrow U \setminus N_{k-1}(x_s, U)$
until $|U| < 3k$
if $|U| \geq 2k$ **then**
  Let $c(U)$ be the centroid of U
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(U))^2$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_r, U)\}$
  $U \leftarrow U \setminus N_{k-1}(x_r, U)$
$\mathcal{C} = \mathcal{C} \cup \{U\}$

An important improvement of MDAV compared to MD is its reduced time complexity. As there is no need to find two elements with greatest distance to each other but only distances of all elements to given vectors, not all distances between any two elements need to be computed. In fact, less than $O(n^2/k)$ distance computations are needed.

Theorem 5.3.
The worst case time complexity of algorithm 2 is $O(n^2/k\ d)$.

Proof. In the first round, $O(nd)$ time is needed to compute the centroid of all elements. Further, $O(nd)$ time is sufficient to find x_r and x_s. By using the selection algorithm, $O(nd)$ is also enough time to find the $k-1$ nearest neighbors to x_r and x_s. In each following round the number of remaining elements is reduced by $2k$. Hence less distances need to be computed in later rounds. As there are $O(n/k)$ rounds the total time complexity of MDAV is $O(n^2/k\ d)$. □

As no distance matrix needs to be maintained, space complexity is trivially bounded by $O(nd)$, which is needed to store the original and anonymized database. As can be seen in the experimental results presented in section 5.2, MDAV is faster and provides information loss comparable to MD.

MDAV+

Another variation of MD is a nameless simplified version of MDAV used as a building block for the V-MDAV algorithm by Solanas and Martínez-Ballesté in [61]. There are four main differences to MDAV.

1. There is no x_s element. In each round only one cluster is created around the element x_r in greatest distance to the global centroid.
2. The global centroid is not updated throughout the algorithm
3. The clustering loop is stopped when there are less than k elements, which are assigned to their closest cluster.
4. All distances are computed and stored beforehand, as in MD.

My research in the context of the MDAV* algorithm published together with Rüdiger Reischuk in [70] showed that the changes described in 1 to 3 deliver equivalent or slightly lowered information losses while performing even faster than MDAV. Hence we introduced the idea as a stand-alone algorithm called MDAV+.

Theorem 5.4.
The worst case time complexity of algorithm 3 is $O(n^2/k\ d)$.

Proof. Despite the claims made in [61], difference 4 does not lower time complexity of the algorithm. As described above, only $O(n^2/k)$ distance computations are needed to find x_r and its $k-1$ nearest neighbors in $O(n/k)$ clustering rounds. Hence, by computing distances on the fly MDAV+ achieves the same time complexity of $O(n^2/k\ d)$ and space complexity of $O(nd)$ as MDAV. □

In practice, MDAV+ is even faster than MDAV as the global centroid is not updated and finding an x_s element is more time consuming as finding another x_r element, given that the distances to the persistent global centroid are stored and sorted beforehand. MDAV+ is not categorized as fixed-size, as like for MD, remaining elements are distributed among several clusters. See section 5.2 for experimental results and algorithm 3 for a detailed description of MDAV+.

Algorithm 3: MDAV$^+$ [61, 70]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
Let $c(X)$ be the global centroid
repeat
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(X))^2$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_r, U)\}$
  $U \leftarrow U \setminus N_{k-1}(x_r, U)$
until $|U| < k$
foreach $x \in U$ **do**
  $clos(x) \leftarrow clos(x) \cup \{x\}$

V-MDAV

The algorithm MDAV creates fixed-size k-member clusterings. Further, MD and MDAV$^+$ assign more than k elements to clusters only on the basis on handling remaining elements and not as an integral part of the clustering process. The algorithm V-MDAV (Variable-sized MDAV) proposed by Solanas and Martínez-Ballesté in [61] combines variable-sized clustering with the maximum distance principle for the first time.

As in MDAV$^+$, a persistant global centroid $c(X)$ is used to find elements x_r in greatest distance to it. After x_r is clustered with its $k-1$ nearest neighbors, a process called *extend* is used to decide whether additional unassigned elements should be clustered with x_r. This way, a cluster including x_r may grow to up to $2k-1$ elements before the process is stopped. Informally, a cluster C is extended by the element $x_I \in U$ which is the unassigned element closest to any element $x \in C$ if and only if $\delta(x, x_I)^2 < \gamma \cdot \delta(x_I, x_O)^2$ for the element $x_O \in U$ being the unassigned element closest to x_I. If x_I is included, the process repeats, otherwise it is halted. A pseudocode of V-MDAV is given as algorithm 4.

The *gain factor* γ dictates, how aggressively the extend process tries to include additional elements. Lets assume $\gamma = 1$. In this case x_I is included in C if x_I is strictly closer to an element in C than to any unassigned neighbor. For $\gamma < 1$ the extension process is less aggressive. To be included anyway, an element must be much closer to an element inside C than to its closest unassigned neighbor. In an extreme case, for $\gamma = 0$ no extension is possible at all. Hence for $\gamma = 0$, V-MDAV computes clusterings identical to those computed by MDAV$^+$. Analogously, for $\gamma > 1$ an element which is closer to its closest unassigned neighbor than to an element in C might be included in C anyway.

To determine an optimal γ for a given database is an unsolved problem. Solanas and Martínez-Ballesté proposed to use $\gamma = 0.2$ for so-called *scattered* databases and $\gamma = 1.1$ for *clustered* databases. However, no criterion is disclosed on how to decide whether a database is scattered or clustered.

Theorem 5.5.
The worst case time complexity of algorithm 4 is $O(n^2 dk)$.

Proof. The additional extension mechanism of V-MDAV comes at the cost of higher computational complexity. Assume that distances are computed on-the-fly, as in our analysis of MDAV$^+$. It takes $O(ndk)$ to find x_I and x_O as well as to decide whether to include x_I in a cluster C. As any cluster might be extended up to $k-1$ times, the worst case time complexity of the extension process is $O(ndk^2)$. Despite the extension there are still $O(n/k)$ clustering rounds V-MDAV uses. Hence, its total worst case time complexity is $O(n^2dk)$, a noticeable slowdown compared to MDAV$^+$. □

Algorithm 4: V-MDAV [61]

input : database X and minimal cluster size k and gain factor γ
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
Let $c(X)$ be the global centroid
repeat
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(X))^2$
  Let $C \leftarrow N_{k-1}(x_r, U)$
  $U \leftarrow U \setminus C$
  repeat
    Let $x_I \leftarrow \arg\min_{y \in U} \min_{x \in C} \delta(x, y)^2$
    Let $\delta_I \leftarrow \min_{x \in C} \delta(x_I, x)^2$
    Let $x_O \leftarrow \arg\min_{y \in U} \delta(x_I, y)^2$
    Let $\delta_O \leftarrow \delta(x_I, x_O)^2$
    if $\delta_I < \gamma \cdot \delta_O$ **then**
      $C \leftarrow C \cup \{x_I\}$
      $U \leftarrow U \setminus \{x_I\}$
    else
      break
  until $|C| = 2k - 1$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
until $|U| < k$
foreach $x \in U$ **do**
  $clos(x) \leftarrow clos(x) \cup \{x\}$

Allowing clusters to have more than k elements is a natural way of improvement for maximum distance heuristics. However, the implementation used in V-MDAV has some drawbacks. Most importantly the greedy nature of the extend process used by V-MDAV ignores the effect the inclusion of an additional element has on clusters to be created later. Even though an element x_I might be closer to the edge of a cluster than to its closet unassigned neighbor x_O, it might still be a bad idea to include x_I because x_O might need to substitute x_I by worse options, if it becomes assigned elsewhere. Another major problem is that V-MDAV is not able to detect natural clusters of size less than k. As shown in figure 5.1, the existence of a close neighbor to x_I does not guarantee that x_I can be clustered with

mostly close neighbors. In an extreme case, x_I and x_O might be close to each other and to C but isolated from other elements. Nevertheless, V-MDAV decides to include neither x_I nor x_O given that X_I and x_O are closer to each other than to the cluster. Some minor disadvantages, namely increased time complexity and the need to find a good parameter γ further hinder the applicability of V-MDAV. More details on parameter influence, time consumption and information loss of V-MDAV are given in section 5.2.

MDAV*

In 2018 together with Rüdiger Reischuk I proposed the algorithm MDAV* [70] as a way to to mitigate the deficiencies of V-MDAV discussed above. As V-MDAV, MDAV* builds upon MDAV$^+$ and adds a mechanism to extend clusters beyond the minimum threshold of k elements. The main novelty of the heuristic MDAV* is to take into account the effects on nearby elements when the extension of a cluster has to be decided and to handle cluster extension before creating a new cluster instead of after the creation. When assigning elements to existing clusters MDAV* considers the additional cost per element (marginal cost) a decision would cause and greedily selects an optimal one.

After the choice of a new cluster origin $x_r \in U$, MDAV* considers two options. The first one is to build a new cluster $N_{k-1}(x_r, U)$ as usual. The second one is to extend the cluster $clos(x_r)$ by x_r in which case the cluster $N_{k-1}(x_r, U)$ cannot be built. Instead, the neighbors of x_r have to be assigned differently. For this, we take the nearest neighbor $\nu(x_r, U)$ of x_r and consider establishing a new cluster around it. The underlying decision rule considers the marginal cost in both cases and chooses the option of lower cost. If marginal cost is equal, a new cluster is created. Still, this is only an estimate of the best possible usage of x_r because we do not know whether $\nu(x_r, U)$ is ever chosen as the origin of a new cluster. For the same reason the cluster around $\nu(x_r, U)$ is not actually created at this time, even if $clos(x_r)$ is extended by x_r.

The cost divided by the number of elements k for creating a new cluster $N_{k-1}(x_r, U)$ out of the element x_r is

$$cost_N(x_r) := \frac{cost(N_{k-1}(x_r, U))}{k}$$

while the cost per element of extending the cluster $clos(x_r)$ by x_r and establishing a new cluster around $\nu(x_r, U)$ (now assigning $k+1$ elements) is

$$cost_E(x_r) := \frac{cost(clos(x_r) \cup x_r) - cost(clos(x_r)) + cost(N_{k-1}(\nu(x_r, U), U \setminus \{x_r\}))}{k+1}.$$

When there are exactly k elements left, $cost_E(x_r)$ cannot be computed. In this case $cost_E(x_r)$ is thought to be infinity so a new cluster $N_{k-1}(x_r, U)$ is created. A parameter γ is not used by MDAV*. A complete description of MDAV* is included as algorithm 5.

Theorem 5.6.
The worst case time complexity of algorithm 5 is $O(n^2\, d)$.

Proof. As before, closest clusters and neighborhoods can be computed in time $O(nd)$. Hence, a single execution of the clustering loop of MDAV* takes $O(nd)$, just like MDAV$^+$.

Algorithm 5: MDAV* [70]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
Let $c(X)$ be the global centroid
repeat
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(X))^2$
  Calculate $cost_N(x_r)$
  Calculate $cost_E(x_r)$
  if $cost_E(x_r) < cost_N(x_r)$ **then**
    $clos(x_r) \leftarrow clos(x_r) \cup \{x_r\}$
    $U \leftarrow U \setminus \{x_r\}$
  else
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_r, U)\}$
    $U \leftarrow U \setminus N_{k-1}(x_r, U)$
until $|U| < k$
foreach $x \in U$ **do**
  $clos(x) \leftarrow clos(x) \cup \{x\}$

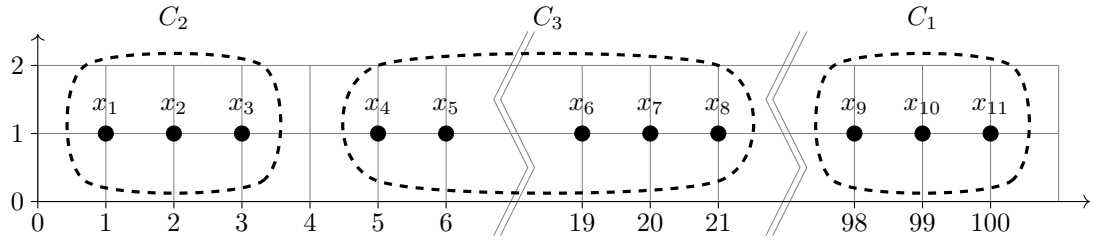
However, in contrast to MDAV$^+$, we cannot guarantee a worst case limit of $O(n/k)$ clustering rounds, as the inclusion of an x_r-element into an existing cluster reduces the size of U by one element only. Thus, there might be up to $O(n)$ clustering rounds and worst case time complexity of MDAV* consequently is $O(n^2\ d)$. □

The example in figure 5.1 shows the difference in clustering behavior between V-MDAV and MDAV*. Both algorithms create the first cluster out of the elements x_9 to x_{11} as x_{11} has maximum distance to the global centroid $c(X) = 34$. Next, x_1 is chosen as x_r-element. In V-MDAV C_2 is created out of x_1 to x_3. It is not extended by x_4, as it has x_5 as a close neighbor for which $\delta_I \geq \gamma \cdot \delta_O$ for all $\gamma \leq 4$. MDAV* on the other hand does include x_4 in C_2. At first, C_2 is created out of x_1 to x_3. Next x_4 is chosen as x_r-element and marginal costs for extension of C_2 and creating a new cluster are evaluated. As $cost_N(x_4) \approx 40.6$ and $cost_E(x_4) \approx 32.19$, C_2 is extended by x_4. Also x_5 is included in C_2, as it becomes x_r element next, $cost_N(x_5) \approx 40.6$ and $cost_E(x_5) \approx 2.6$. As a result, in this scenario the total clustering cost is significantly lower for MDAV*.

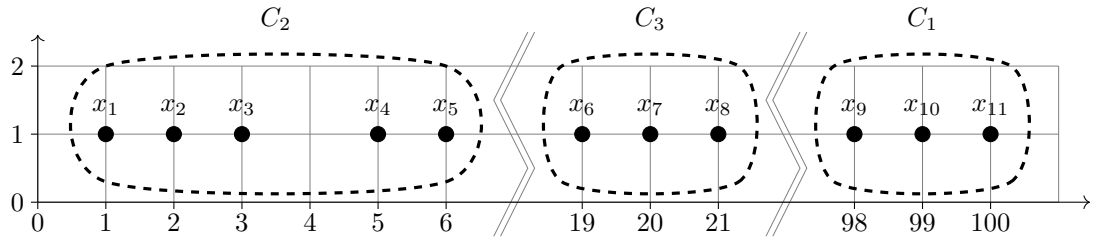
Beyond MDAV*

Since the original publication in 2018 I have designed and tested several yet unpublished improvements to MDAV* which are able to further lower information loss, as experimental data provided in section 5.2 shows. For the purpose of this dissertation this adapted MDAV* heuristic is called MDAV$^*_\gamma$.

The first improvement is the re-introduction of a gain factor γ to adjust aggressiveness



(a) Clustering $\mathcal{C}_1$ by V-MDAV



(b) Clustering $\mathcal{C}_2$ by MDAV*

Figure 5.1: Example database $X \in \mathcal{X}_{11\times 2}$ in which V-MDAV clusters worse than MDAV*, because for all $\gamma \leq 4$ V-MDAV does not detect that x_4 should be included in C_2. The global centroid is $c(X) = 34$ and costs are $cost(\mathcal{C}_1) = 258.8$, $cost(\mathcal{C}_2) = 21.2$.

of extension for different databases. Like for V-MDAV a gain factor of $\gamma = 0$ causes the algorithm to behave like MDAV$^+$, whereas for $\gamma = 1$ the algorithm behaves like MDAV*. Again, values in between or above 1 are possible as well.

A second improvement made in cooperation with Mustafa Sahin added another extension criterion, which allows to extend a cluster C by an element x_r even if $\nu(x_r, U)$ cannot be clustered cost effectively without x_r. The additional criterion checks whether marginal cost of x_r and $\nu(x_r)$ when clustered with C are lower than the marginal cost of elements in the original cluster C:

$$cost_{E2}(x_r) := \frac{cost(clos(x_r) \cup \{x_r, \nu(x_r, U)\}) - cost(clos(x_r))}{2}$$

This check is applied when the extension rule of MDAV* opts for the creation of a new cluster and is especially beneficial in situations in which either both x_r and $\nu(x_r, U)$ or none of them should be included in C to minimize cost. As before, when there are exactly k elements unassigned at the time of choosing x_r, a new cluster will be created. Note that even in the case $cost_{E2}(x_r) < cost_N(x_r)$ the cluster $clos(x_r)$ is extended by x_r only. This approach is aimed at considering uncertainty about future clusters. Not including $\nu(x_r, U)$ right away allows it to be clustered somewhere else, when it is indeed not chosen as a new x_r element later on. However, if it is chosen as x_r element, a new decision with more accurate data can be done. In general, it is save to assume that delaying any extension decisions as long as possible is safer, as these decisions relay on assumptions on the future clustering process, which might not be accurate.

Another design decision is the application of the gain factor only for the original extension check. As the original and new extension check consider different scenarios, different

gain factors will be optimal for each of them on a given database. However, introducing a second gain factor γ' would increase the amount of work to be done to tune parameters disproportionately. Hence, I opted to omit a second gain factor altogether.

There is no difference in time or space complexity between MDAV* and MDAV$^*_\gamma$ besides the extra time needed to find a suitable γ for the database given. A pseudocode of MDAV$^*_\gamma$ is given as algorithm 6.

Algorithm 6: MDAV$^*_\gamma$

input : database X and minimal cluster size k and gain factor γ
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
Let $c(X)$ be the global centroid
repeat
  Let $x_r \leftarrow \arg\max_{x \in U} \delta(x, c(X))^2$
  Calculate $cost_N(x_r)$
  Calculate $cost_E(x_r)$
  Calculate $cost_{E2}(x_r)$
  if $cost_E(x_r) < \gamma \cdot cost_N(x_r)$ ***or*** $cost_{E2}(x_r) < cost_N(x_r)$ **then**
    $clos(x_r) \leftarrow clos(x_r) \cup x_r$
    $U \leftarrow U \setminus \{x_r\}$
  else
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{N_{k-1}(x_r, U)\}$
    $U \leftarrow U \setminus N_{k-1}(x_r, U)$
until $|U| < k$
foreach $x \in U$ **do**
  $clos(x) \leftarrow clos(x) \cup \{x\}$

When considering the possibility of cluster extension by two elements at once, one might be tempted to generalize this behavior to consider the extension of up to $k-1$ elements at once, depending on the size of $clos(x_r)$. Despite the obvious increase in time complexity there is another good reason not to pursue this approach: Deciding for a cluster extension or cluster generation inherently uses assumptions about future clustering processes. When more and more elements are included in this decisions, we also need to assume more about their ideal clustering, all without being able to see the big picture of clustering opportunities much later during the run of the algorithm. Hence, there is not only diminishing return in including more elements in our extension check but also a negative impact of a decision based on weak data preventing a simple design of even smarter extension rules.

Determination of the Closest Cluster

As discussed in the introduction of this chapter, the closest cluster C to an element x is assumed to be

$$clos(x) := \arg\min_{C \in \mathcal{C}} \delta(x, c(C))^2$$

for all heuristics so far. Another solution to this problem would be to consider cluster C which contains the clustered element y closest to x, i.e.

$$clos'(x) := \arg\min_{C \in \mathcal{C}} \min_{y \in C} \delta(x, y)^2.$$

This operation would be similar to the computation of x_I in `V-MDAV`. A third option proposed in [70] is to choose the cluster C which has the minimal cost increase, when extended with x among all current clusters, i.e.

$$clos''(x) := \arg\min_{C \in \mathcal{C}} cost(C \cup x) - cost(C).$$

As $clos(\cdot)$ is used to extend clusters and our main goal is to minimize information loss, the third option seems to be the obvious choice. Fact 5.7 and corollary 5.8 further indicate that choosing the closest cluster according to strategy 1 or 2 can indeed increase cost compared to the third option.

Fact 5.7.
The distance measures between a cluster C and an element x used by $clos(x)$ and $clos'(x)$ depend on the group size of C.

Proof. Consider a cluster C containing k elements, with equal attributes. Further there is an element x with (Euclidean) distance of d to the elements of C. When including x in C, the centroid of C shifts towards x and so $cost(C)$ increases from 0 to

$$k \cdot \left(\frac{d}{k+1}\right)^2 + \left(d - \frac{d}{k+1}\right)^2 = d^2 \frac{k}{k+1}.$$

□

Corollary 5.8.
A smaller cluster C can be a better extension partner for x, even if there is another cluster C closer to it according to $clos(x)$ or $clos'(x)$.

However, experimental analysis depicted in table 5.7 indicate that at least for maximum distance heuristics there is a significant slowdown without adequate information loss reduction of $clos''(\cdot)$ compared to $clos(\cdot)$. Hence, no further effort is made to include $clos''(\cdot)$ into these heuristics.

Overview

Table 5.2: Key properties of all maximum distance heuristics presented in this section.

	Time	Space	Variable-Size	Gain Factor	Introduced	Reference
MD	$O(n^2 \cdot \max\{n/k, d\})$	$O(n^2)$	yes	no	1998	[51]
MDAV	$O(n^2/k\ d)$	$O(nd)$	no	no	2005	[20]
MDAV$^+$	$O(n^2/k\ d)$	$O(nd)$	yes	no	2006 (2018)	[61], [70]
V-MDAV	$O(n^2k\ d)$	$O(nd)$	yes	yes	2006	[61]
MDAV*	$O(n^2\ d)$	$O(nd)$	yes	no	2018	[70]
MDAV$^*_\gamma$	$O(n^2\ d)$	$O(nd)$	yes	yes	-	-

5.2 Experimental Evaluation of Maximum Distance Heuristics

To compare different heuristics several real and synthetic benchmark databases have been used, in particular *Census*, *Tarragona* and *EIA* from the CASC project [17], *Cloud1*, *Cloud2*, the *Adult data set* and the *credit card clients data set* from the UCI Machine Learning Repository [42] as well as uniformly distributed synthetic databases *SimU* and clustered synthetic databases *SimC* created as proposed in [16].

The Census database contains 1080 elements in 13 numerical attributes. It was created using the Data Extraction System of the U.S. Bureau of Census in 2000. Tarragona contains 834 elements in 13 numerical attributes. It contains company data from the Spanish region Tarragona from 1995. The EIA (Energy Information Authority) data set consists of 4092 elements in 15 attributes. As in previous works (see e.g. [16]) only a subset of 11 numeric attributes precisely 1 and 6 to 15 has been used. Cloud1 and Cloud2 have been created using statistics from *AVHRR* images. These databases are commonly used for the training and evaluation of machine learning algorithms, and both contain 1024 elements with 10 attributes each. Adult consists of 48842 elements with 14 attributes. As for EIA, only a subset of numerical attributes namely *age*, *education number* and *hours per week* is used. This particular selection was suggested in [58]. It is used to test algorithms on low-dimensional data. The Credit Card clients data set consists of 30000 elements in 24 numeric attributes. It contains inputs and predictive results from six data mining methods and is used to evaluate algorithms on higher-dimensional data.

For experiments with uniformly distributed synthetic data, databases SimU_0 to SimU_{24} were created. Each of them consists of 1000 elements in 10 independently chosen numerical attributes. When reporting results on SimU, averages over all 25 databases are used. Synthetic clustered data is tested in the same way. 25 databases SimC_0 to SimC_{24} are created and results are reported as averages over information losses and runtimes on all SimC variants. To create a SimC database the probabilistic procedure depicted in algorithm 7 is used. In summary, 100 clusters with between 4 and 21 elements in close distance are created and mixed with some additional unrelated (noise) elements.

For a meaningful test the attributes of the databases are standardized to mean value 0 and variance 1 prior to anonymization. This ensures that all dimensions have equal im-

Algorithm 7: Creation of a SimC database [16]

output: Database X

Let $X \leftarrow \emptyset$
for $i \in \{0, \ldots, 99\}$ **do**
    Draw element c_i randomly from $[-10000, 10000]^{10}$
    $X \leftarrow X \cup \{c_i\}$
foreach $c_i \in \{c_0, \ldots, c_{99}\}$ **do**
    Let $r \in_R \{3, \ldots, 20\}$
    for $j \in \{1, \ldots, r\}$ **do**
        Draw elements x_i^j randomly from $[-50, 50]^{10}$
        $x_i^j \leftarrow c_i + x_i^j$
        $X \leftarrow X \cup \{x_i^j\}$
Let $n = |X|$
for $i \in \{0, \ldots, \lfloor n/3 \rfloor\}$ **do**
    Draw element y_i randomly from $[-10000, 10000]^{10}$
    $X \leftarrow X \cup \{y_i\}$

pact on the anonymization process and information loss evaluation. As microaggregation is dimension and order conserving, this standardization can be reversed after anonymization and does not introduce further distortion. All information losses given in this evaluation are expressed in percentages to be directly comparable with previously published results, see definition 2.23. Computations have been performed based on single-threaded Java implementations on a PC equipped with an AMD Ryzen 9 5900X with 4.8 GHz turbo frequency and 32 GB of DDR4-3200 MHz RAM.

Results

All algorithms are benchmarked against each other in the databases described above. For each database experiments were made with all $k \in \{2, 3, 4, 5, 7, 10\}$ to evaluate information loss and performance for different anonymity guarantees. Because no simple and reliable way of determining a good value of γ for the use with `V-MDAV` or $\texttt{MDAV}^*_\gamma$ is known, values of 0 to 2 in steps of 0.1 are used for γ and information loss and time consumption is reported only for the best result, i.e. lowest information loss, observed. Table A.9 in the appendix summarizes the γ values used. All algorithms are compared in regards of information loss and runtime. Further, average cluster sizes are compared for all variable-size algorithms. For further results omitted in this section consult appendix A.

As expected, increasing k also increases information loss for any maximum distance heuristic. Further, the differences between information losses on different databases outweighs the differences between different heuristics evaluated. There seem to be databases that are more resistant to anonymization than others. This interpretation is further supported by the fact that no algorithm is consistently better than another algorithm on one particular database, but consistently worse in another.

Ignoring time and space requirements, it can be observed that neither `MDAV` nor $\texttt{MDAV}^+$

are able to obtain better results than MD on average. Further, there seems to be little influence of the security parameter k on which of these three algorithms performs best. Despite the fact that V-MDAV levitates variable-size mechanisms, has higher time complexity and is evaluated for the best out of 21 gain factors γ, it delivers only a slight reduction in information loss compared to MDAV$^+$ on average. This slight reduction is indeed remarkable low as we allow $\gamma = 0$ for our test, so V-MDAV can by design never be worse than MDAV$^+$. In 11 out of the 54 tests (combinations of different databases and anonymity parameters k) V-MDAV is not able to use cluster extension at all, to reduce information loss. Hence, in these cases $\gamma = 0$ is chosen. As our experiments show, MDAV* is able to outperform MDAV even without the use of gain factors adjusted to a particular database. On average it achieves about 7% and 4% lower information loss compared to MDAV$^+$ and V-MDAV respectively. Adding an adjustable gain factor and improved cluster extension mechanisms further improves MDAV* significantly. By allowing $\gamma = 0$ as well as $\gamma = 1$ it is further guaranteed that the algorithm can never introduce information losses above those of MDAV$^+$ or MDAV*. In none of the performed tests a gain factor of $\gamma = 0$ resulted in best information loss for MDAV$^*_\gamma$. Furthermore, optimal gain factors for MDAV$^*_\gamma$ are quite consistent, ranging from 0.6 to 1.1 over all test cases compared to a range of 0 to 1 for V-MDAV. This shows that the cluster extension mechanism of MDAV* and MDAV$^*_\gamma$ is superior and easier to tune than that of V-MDAV.

The high time consumption of MD is as expected, given its much higher asymptotic complexity, compared to newer heuristics. Between MDAV and MDAV$^+$ which have the same asymptotic time complexity, MDAV$^+$ seems to profit more from larger values of d as e.g. found in Credit Card for which it has less than half the time consumption of MDAV. Nevertheless, even for low dimensional data like Adult, MDAV$^+$ is still notably faster likely due to the persistent global centroid as discussed in the previous section. The variable-size mechanics of V-MDAV and MDAV* come at the cost of higher runtimes. However, the real-world performance of V-MDAV does not seem to match its theoretical worst case time complexity of $O(n^2kd)$ derived above. Instead of a linear increase in time consumption with increased k, the algorithm becomes slightly faster for greater values of k. This behavior can be explained by looking at the average cluster size and γ used (e.g. table A.9) to achieve optimal results in our experiments. It can be observed, that only very few cluster extensions are made, given they are profitable at all. As extending clusters is, per element, much more time consuming than creating new clusters, the worst case time complexity of V-MDAV is not as relevant in practice.

A similar, although weaker form of this behavior can be observed for MDAV* and MDAV$^*_\gamma$ which are not slower than e.g. MDAV$^+$ by a factor of k. Unlike implied by the asymptotic worst case time complexity of $O(n^2d)$ the heuristics are getting faster with larger values of k. Both can be explained by the unusual high amount of cluster extension assumed for computing the worst case complexity. However, as the average cluster sizes and used gain factors are significantly higher than those of V-MDAV the new algorithms are in most cases slower than V-MDAV.

Comparing not only information loss or performance isolated from each other but general overall behavior instead, there is no clear *best* algorithm. When time is critical, MDAV$^+$ should be considered, as it is the fastest of these algorithms while delivering results comparable to MD and MDAV which are much slower. When an addition factor of 2 in runtime is acceptable, but there is no time to evaluate good gain factors, MDAV* is a good choice.

However, if time consumption is secondary, the preferred choice should be MDAV$^*_\gamma$ as it is guaranteed to produce lower or equal information losses than MDAV$^+$ and MDAV*. Moreover, as our experiments show, is not likely to perform worse than MD MDAV or V-MDAV on any real world databases. Applying the original MD or MDAV heuristics should be avoided as there seems to be no situation in which their increased time and space requirements result in any meaningful advantage above newer maximum distance heuristics.

Table 5.3: Average percentage information loss difference between maximum distance heuristics over all benchmarks. A value below 100 indicates that an algorithm causes less information loss on average. Differences based on MD do not include data from the Adult database, as it is too big to be handled with quadratic space requirements.

Information Loss Differences in %						
from\to	MD	MDAV	MDAV$^+$	V-MDAV	MDAV*	MDAV$^*_\gamma$
MD	100	100.3	100.2	96.6	93.9	91.8
MDAV		100	99.9	96.2	92.6	89.8
MDAV$^+$			100	96.3	92.7	89.9
V-MDAV				100	96.2	93.1
MDAV*					100	96.7
MDAV$^*_\gamma$						100

Table 5.4: Information losses, runtimes, average cluster sizes and used gain factors γ of maximum distance heuristics on EIA $\in \mathcal{X}_{4092,11}$ from the CASC project. Average cluster sizes are reported for variable-size heuristics only.

	Information Loss in % on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	0.27	0.47	0.77	1.67	2.28	3.71
MDAV	0.31	0.48	0.67	1.67	2.17	3.84
MDAV$^+$	0.32	0.49	0.67	1.78	2.21	3.55
V-MDAV	0.23	0.46	0.67	1.06	2.21	2.79
MDAV*	0.22	0.45	0.62	0.91	2.03	2.63
MDAV$^*_\gamma$	0.20	0.39	0.54	0.82	1.66	2.18

	Runtime in s on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	2.38	1.58	1.15	0.91	0.67	0.49
MDAV	0.23	0.15	0.10	0.09	0.07	0.06
MDAV$^+$	0.11	0.06	0.05	0.04	0.03	0.02
V-MDAV	0.15	0.13	0.10	0.11	0.08	0.13
MDAV*	0.37	0.24	0.18	0.14	0.10	0.09
MDAV$^*_\gamma$	0.30	0.22	0.18	0.18	0.18	0.14

	Average cluster size on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.17	3.61	4.00	5.54	7.01	11.46
MDAV*	2.24	3.38	4.24	5.62	7.56	10.88
MDAV$^*_\gamma$	2.19	3.37	4.60	6.14	9.39	12.00

	γ used on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.2	0.6	0.0	0.4	0.0	1.3
MDAV$^*_\gamma$	0.8	0.7	0.8	1.2	1.1	1.2

Table 5.5: Information losses, runtimes, average cluster sizes and used gain factors γ of maximum distance heuristics on Credit Card $\in \mathcal{X}_{30000,24}$ from the UCI repository. Average cluster sizes are reported for variable-size heuristics only.

	Information Loss in % on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	3.66	6.43	8.56	10.23	12.64	15.16
MDAV	3.66	6.40	8.53	10.17	12.58	15.18
MDAV$^+$	3.65	6.39	8.49	10.17	12.59	15.21
V-MDAV	3.64	6.38	8.49	10.17	12.58	15.17
MDAV*	3.65	6.44	8.48	10.22	12.36	14.67
MDAV$^*_\gamma$	3.59	6.25	8.25	9.83	12.11	14.47

	Runtime in s on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	895.1	590.3	441.1	354.6	256.0	182.9
MDAV	16.4	11.8	9.4	8.6	6.2	5.3
MDAV$^+$	6.7	4.6	3.3	2.7	2.0	1.5
V-MDAV	15.4	11.4	9.9	8.9	8.1	7.3
MDAV*	24.5	17.2	14.4	12.7	10.8	9.2
MDAV$^*_\gamma$	28.2	19.3	16.0	13.7	12.1	9.8

	Average cluster size on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
V-MDAV	2.02	3.02	4.00	5.00	7.03	10.04
MDAV*	2.21	3.50	4.75	6.01	8.45	11.94
MDAV$^*_\gamma$	2.11	3.23	4.35	5.46	7.99	11.30

	γ used on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
V-MDAV	0.1	0.1	0.0	0.0	0.1	0.1
MDAV$^*_\gamma$	0.8	0.8	0.8	0.8	0.9	0.9

Table 5.6: Mean information losses, runtimes and used gain factors γ of maximum distance heuristics on sets of synthetic databases SimC $= \{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$. Average cluster sizes are reported for variable-size heuristics only.

	Mean Information Loss in % on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	3.88	7.00	9.32	11.33	14.51	18.37
MDAV	3.86	6.93	9.32	11.26	14.46	18.45
MDAV$^+$	3.82	6.86	9.23	11.04	14.11	18.09
V-MDAV	3.32	6.19	8.35	10.16	13.14	16.97
MDAV*	3.36	6.00	8.17	9.66	12.32	15.69
MDAV$^*_\gamma$	3.28	5.73	7.61	9.16	11.65	14.84

	Mean Runtime in s on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	0.12	0.08	0.06	0.05	0.04	0.03
MDAV	0.02	0.02	0.01	0.01	0.01	0.01
MDAV$^+$	0.01	0.01	0.01	0.01	0.00	0.00
V-MDAV	0.02	0.02	0.02	0.02	0.02	0.02
MDAV*	0.05	0.03	0.03	0.02	0.02	0.02
MDAV$^*_\gamma$	0.05	0.03	0.03	0.03	0.03	0.02

	Mean γ used on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
V-MDAV	0.2	0.4	0.5	0.5	0.7	0.8
MDAV$^*_\gamma$	0.8	0.8	0.8	0.8	0.9	1.1

Table 5.7: Results of MDAV* with alternative *clos* procedure on Credit Card

	Information Loss in % on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MDAV*	3.65	6.44	8.48	10.22	12.36	14.67
MDAV* WITH $\mathit{CLOS}''(\cdot)$	3.65	6.44	8.48	10.21	12.36	14.68

	Runtime is s on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MDAV*	24.5	17.2	14.4	12.7	10.8	9.2
MDAV* WITH $\mathit{CLOS}''(\cdot)$	228.0	135.1	99.7	80.8	60.8	49.2

6

Lloyd-based Heuristics

MDAV and its variants are considered to be the most popular heuristics for k-anonymous microaggregation. There are, however, many alternatives that use other principles to achieve the same goal. The topic of this chapter is the so-called Lloyd-based principle which is named after the famous k-means clustering algorithm by Stuart Lloyd [45].

Adapting the round-based optimization process of Lloyd's algorithm to create k-member clusterings is no easy task as there are quite a lot of differences between both problems, as discussed in section 4.1. Therefore, Lloyd-based heuristics are more diverse and complex than MDAV-variants. Nevertheless, applying a Lloyd-like cluster optimization strategy has been an obvious way to move forward in achieving microaggregation with lower information loss, due to its success for k-means clustering.

Before going into more detail about the different approaches, we first recap Lloyd's algorithm and an important improvement called k-MEANS++. After that, we take a short look at an early Lloyd-based approach called PCL. Next, a post-processing approach called LMaS is discussed in detail. The last approaches discussed in this chapter are ONA and its recent improvement ONA* which are state-of-the-art Lloyd-based heuristics able to outperform any MDAV-variant with negligible computational overhead. Finally, LMaS, ONA and ONA* are benchmarked against MDAV$^+$ and MDAV* to set their performance into perspective.

6.1 Lloyd's Algorithm

As defined in section 4.1, the concept of k-means clustering is similar to that of a k-member clustering used for k-anonymous microaggregation and is commonly used in the field of data mining. As for microaggregation, the goal of k-means clustering is to partition elements in an d-dimensional space into clusters. Further, the quality of a partitioning is measured by the same distortion cost function using the sum of squared errors. However, unlike k-anonymous microaggregation, k-means clustering does not guarantee a minimum cluster size, instead a fixed (maximum) number of clusters has to be achieved (see definitions 4.1 and 4.2).

Lloyd's algorithm (abbreviated as LLOYD in the upcoming text) takes a database X and a value k and creates a k-clustering $\mathcal{C}(X)$. The result is a valid solution to the k-means clustering problem but not necessarily optimal. At first, k cluster centers are chosen.

The standard procedure called *Forgy method*[3] randomly chooses existing elements from the database. Unlike the primitive method of creating artificial data vectors as initial centers, this behavior ensures that initial cluster centers are in populated areas of the data space and further that no empty clusters are created throughout the process.[4] The alternative *random partition method* assigns each element to a random center and creates the initial cluster centers as the centroids of the resulting clustering.

After initialization the algorithm proceeds with two alternating steps. In the *assignment step* each element is assigned to its closest cluster center using the squared Euclidean distance. Afterwards, in the *update step* the cluster centers are moved to the centroids of the clusters resulting from the previous assignment step. These steps are repeated until a convergence condition it met. Convergence of the algorithm is not guaranteed due to numeric imperfections and the possibility of toggling between equally good solutions. Even if the algorithm achieves convergence, it will in general only find a local optimum. To guarantee convergence in practice, the convergence criterion is chosen as a combination of a bound on the difference between cost of the previous and current solution as well as a hard round limit. Due to the probabilistic behavior of initial center generation, the algorithm is usually executed several times to obtain a list of clusterings to choose from. A pseudocode of LLOYD is given as algorithm 8.

Algorithm 8: LLOYD (Lloyd's algorithm) [45]

input : database X and an integer k
output: k-clustering $\mathcal{C} = \{C_1, \dots, C_k\}$

```
Let C = {C_1, ..., C_k} ← {∅, ..., ∅}
Create initial centers c_1, ..., c_k
repeat
    foreach x ∈ X do
        Let i ← arg min_{1≤j≤k} δ(x, c_j)^2
        C_i ← C_i ∪ {x}
    for 1 ≤ i ≤ k do
        c_i ← c(C_i)
until convergence
```

The intuition of LLOYD is to alternate between two optimization goals to achieve a locally minimal distortion cost. On the one hand, in an optimal k-clustering and for given cluster centers, any element should be assigned to its closest cluster center. On the other hand, for given clusters, the center of any cluster should be the centroid of its elements. Alternating these goals changes clusters and centers until a solution, which satisfies both conditions, is created. However, optimality of the solution is not guaranteed, as both conditions are necessary but not sufficient for optimal solutions.

It should be obvious that Lloyd's algorithm cannot exceed the given limit on the number of clusters k. It is allowed, but not beneficial to create empty clusters during the process of

[3] Named after its author E. Forgy.
[4] For the second condition to hold, it must be guaranteed that no two initial centers are chosen as elements with identical attribute vectors.

k-means clustering. When a cluster becomes empty after a reassignment step, the cluster becomes lost, as no centroid can be computed and as a consequence, no more elements will be assigned to the cluster. Hence, such situations need to be avoided, e.g. by using the Forgy initialization.

Theorem 6.1.
Assuming i is the total number of rounds until convergence, the worst case time complexity of algorithm 8 is $O(ndki)$.

Proof. The time complexity of LLOYD depends on several factors. In each round $O(nkd)$ time is needed to find the closest cluster center to all elements. Further, it takes $O(nd)$ to update all centroids. Assuming initialization and checking the convergence criterion is cheap, the total time complexity is $O(ndki)$, where i is the total number of rounds until the algorithm halts. □

Of course it is of interest to further specify i. However, as discussed above, without a hard round limit, the algorithm might never halt. Hence, a hard round limit is currently the only way of specifying i.

k-means++

The main disadvantage of LLOYD is its lack of quality guarantees on the resulting clusterings. In 2006, Arthur and Vassilvitskii proposed a major improvement called k-MEANS++ [4] fixing this problems without sacrifices in practical data quality or time complexity. By using a new initialization method, k-MEANS++ is able to guarantee the expected distortion cost on any database X to be smaller or equal than $8(\ln k + 2)$ times the distortion cost of an optimal k-means algorithm on X, i.e.:

Theorem 6.2 ([4]).
For any database X and an optimal k-means clustering $\mathcal{C}_{OPT}(X)$ according to definition 4.1 the expected cost $E[cost(\mathcal{C}(X))]$ of a clustering $\mathcal{C}(X)$ created by k-MEANS++ is bounded by

$$E[cost(\mathcal{C}(X))] \leq 8(\ln k + 2)\, cost(\mathcal{C}_{OPT}(X)).$$

As for the forgy method, initial cluster centers are elements $x \in X$. The novelty of k-MEANS++ is that centers are chosen with higher probability when they are in greater distance to elements already chosen.

After the first center c_1 is chosen at random, for any element x the minimum (Euclidean) distance $D(x)$ to any existing center c is computed. After that, an element x is chosen as the next center by using the probability distribution defined by

$$p(X) = \frac{D(x)^2}{\sum_{x \in X} D(x)^2}.$$

The updating of $D(\cdot)$ and the selection of centers c_i continues, until k centers are chosen. After the initialization is done, LLOYD is executed as usual, beginning at step 3 of algorithm 8. See algorithm 9 for a formal description of the new initialization process.

Algorithm 9: k-MEANS++ initialization [4]

input : database X and an integer k
output: Set of initial centers $C = \{c_1, \ldots, c_k\}$

Draw c_1 randomly from X
Let $C \leftarrow \{c_1\}$
for $2 \leq i \leq k$ **do**
  foreach $x \in X$ **do**
    Let $D(x) \leftarrow \min_{c \in C} \delta(x, c)$
  Draw c_i from X by choosing $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
  $C \leftarrow C \cup \{c_i\}$

The approximation factor of $O(\log k)$ is proven to be valid after initialization and clustering every element with its closest cluster. As further steps of LLOYD cannot increase cost, the guarantee is still valid when the algorithm halts. The proof further uses the known cost increase that occurs when another element than the centroid is used as center (see theorem 2.15, which is valid for k-means clustering as well) and the fact that in any k-means clustering any element can and should be clustered with its closest cluster center (see theorem 4.3).

On a high level, it is shown that the cost of the initial clustering $\mathcal{C}$ created by k-MEANS++ is lower or equal to 8 times the cost of an optimal clustering $\mathcal{C}_{OPT}$, assuming a center is chosen from every optimal cluster $C \in \mathcal{C}_{OPT}$. It is further argued, that the chances of a cluster $C \in \mathcal{C}_{OPT}$ not being represented by a center is bounded by $O(\log k)$. The full proof can be found in the original publication (see [4]).

Unfortunately, the original paper does not give complexity bounds. However, assuming efficient implementation it can be shown that the asymptotic time complexity of k-MEANS++ is not increased compared to LLOYD with traditional initialization.

Theorem 6.3.
The worst case time complexity of algorithm 9 is $O(ndk)$.

Proof. Calculating $D(x)$ takes $O(ad)$, assuming there are a cluster centers already chosen. Instead of recalculating $D(x)$ from scratch after a new center c is chosen, it can be updated by computing $\delta(x, c)$ only and comparing it to the previous value of $D(x)$. In summary, $O(k)$ centers each require n distance computations of complexity $O(d)$. Hence, throughout the initialization it takes time $O(ndk)$ to calculate and maintain $D(\cdot)$. The sum of minimum distances can be computed and updated analogously within the same time complexity bound.

To efficiently choose a new center according to $p(X)$, assume a total ordering $x_1, \ldots, x_n$ of elements $x \in X$ and randomly draw a number r between 0 and $\sum_{x \in X} D(x)^2$. The element x_i becomes the new center if $\sum_{j=1}^{i-1} < r \leq \sum_{j=1}^{i}$. Maintaining and updating the prefix sums takes $O(n)$ for every new center chosen. So the total time complexity of random center selection is $O(nk)$. In total, the time complexity of k-MEANS++ initialization is $O(ndk)$ which is majorized by the complexity of $O(ndki)$ of performing Lloyd-like cluster optimization rounds. □

6.2 Probability-Constrained Lloyd

In 2013 Rebollo-Monedero et al. introduced the microaggregation heuristic PCL (Privacy-constrained Lloyd) [58]. The main concept of the heuristic is to alter the optimization goal of the assignment step from LLOYD to force it to create clusters of size at least k. To explain the details of PCL a change of perspective is needed, as the algorithm assumes X to be a random variable defined by the empirical distribution $p_X(x)$ of elements x in a given database of n elements.

k-anonymity is expressed as probability constraints $p_0(C) = \frac{1}{\lfloor n/k \rfloor}$ that apply to each cluster C of the resulting clustering $\mathcal{C}$ and ensure $np_0(C) \geq k$. Membership of elements in clusters is expressed by a probability mass function $p_{\mathcal{C}}(C)$, for which $p_{\mathcal{C}}(C) = p_0(C)$ is required. The update function is identical to that of LLOYD, as centroids should be used for both clustering principles. Further, initialization and convergence can be handled as in LLOYD, as well.

A cost function $c_k : \mathcal{C} \to \mathbb{R}$ is found and computed for every cluster C to ensure $p_{\mathcal{C}}(C) = p_0(C)$. Elements x are assigned to clusters C_i with center c_i with

$$i = \arg \min_{1 \leq j \leq \lfloor n/k \rfloor} \delta(x, c_j)^2 + c_k(C_j).$$

Besides performing the usual tasks of LLOYD, PCL needs to compute and update $c_k(\cdot)$ to achieve a valid k-member clustering. However, as the authors admit, this may not be possible for discrete probability distributions of X. Instead, a method assuming continuous data and applying algorithms to solve systems of nonlinear equations is presented and relaxed to handle (large) finite databases as well. Unfortunately, this leads to serious problems for $k < 500$ and making PCL practically infeasible for microaggregation with $k < 100$. See [58] for more details on how to compute c_k. A pseudocode of PCL is included as algorithm 10.

Algorithm 10: PCL [58]

input : database X of size $n = |X|$ and an minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $\mathcal{C} = \{C_1, \ldots, C_{\lfloor n/k \rfloor}\} \leftarrow \{\emptyset, \ldots, \emptyset\}$
Create initial centers $c_1, \ldots, c_{\lfloor n/k \rfloor}$
Let initial cost $c_k(C) \leftarrow 0$ for all $C \in \mathcal{C}$
repeat
    Adjust $c_k(\cdot)$ to satisfy $p_{\mathcal{C}}(C) = p_0(C)$ for each current $C \in \mathcal{C}$
    foreach $x \in X$ **do**
        Let $i \leftarrow \arg\min_{1 \leq j \leq \lfloor n/k \rfloor} \delta(x, c_j)^2 + c_k(C_j)$
        $C_i \leftarrow C_i \cup \{x\}$
    for $1 \leq i \leq \lfloor n/k \rfloor$ **do**
        $c_i \leftarrow c(C_i)$
until *convergence*

For an experimental evaluation the authors used k between 500 and 4000 for the Adult database (see 5.2 for a definition) and $k \geq 1024$ for experiments with 65536 Gaussian points in 2 to 4 dimensions.

While no exact asymptotic computational complexity of PCL is stated by its authors, their experiments show that it is likely to be quadratic in n. The constants, however, seem way higher for PCL compared to MDAV variants. In situations ($k = 1000$, n between 30000 and 100000) in which MDAV takes between $0.1s$ and $1.5s$ to compute its output, PCL takes between $70s$ and $800s$.

Due to the lack of implementation details and non-applicability for small databases or reasonable values of k, PCL has not been implemented during the work for this dissertation. However, looking at the results provided by the authors of PCL, it is clear that an adaption to LLOYD can give better results than a greedy heuristic like MDAV, at least in situations with large k, n and a generous time budget. As we will see later in section 6.4, there is indeed another way to adapt LLOYD resulting in excellent results, without the compromises made by PCL.

6.3 Merge and Split Approach

Before we go into details on more modern approaches to adapt LLOYD, this section covers a post-processing approach designed during my work for this dissertation. Although this approach is still unpublished, due to the rise of native adaptions of LLOYD in recent years, it is still worth mentioning what can be achieved just by adapting the results of LLOYD. The basic idea of LMaS (Lloyd, Merge and Split) is simple: LLOYD or k-MEANS++ is executed on a database X to obtain $\lfloor n/k \rfloor$ clusters. Now clusters with less than k elements are merged with neighboring clusters to obtain a k-member clustering. Afterwards clusters that became larger than $2k$ elements are split by applying a simple microaggregation heuristic like MDAV$^+$.

In an ideal scenario, the clusters generated by an $\lfloor n/k \rfloor$-means algorithm would be close to those of a k-member clustering for the same utility goal of low information loss. Hence, as LLOYD creates excellent results for k-means clustering, we could expect only a small number of merge and split operations necessary and thus obtain a good solution for k-anonymous microaggregation. In practice it can be observed that given the right choice of parameters LMaS is able to match or even outperform MDAV$^*_\gamma$ in terms of information loss. However, due to the probabilistic behavior of Lloyd's algorithm, the time needed to find such a solution is significantly higher.

A general overview of LMaS is given as algorithm 11. There are three main steps that make up the algorithm. First, k-MEANS++ (see algorithm 9) is applied to find an $\lfloor n/k \rfloor$-means clustering. Next, the algorithm MERGE (see algorithm 12) is used to enforce a minimum cluster size of k. Lastly, the SPLIT algorithm depicted in algorithm 13 is used to minimize information loss by reducing the average cluster size without violating the k-member requirement. As a probabilistic initialization is involved, LMaS should be executed several times to obtain good solutions. However, it should be noted that MERGE and SPLIT are deterministic, so the only difference between different runs of LMaS is caused by the differences in the sets of initial centers created by the k-MEANS++ initialization.

Theorem 6.4.
Given $k \leq \sqrt{n}$, the worst case time complexity of algorithm 11 is $O(n^2dik^{-1})$.

Proof. The first step of LMaS takes $O(n^2dik^{-1})$, as this is the time complexity of k-MEANS++ for $\lfloor n/k \rfloor$ clusters on d-dimensional databases of size n. In the MERGE step, there are at most $O(n/k)$ merge operations until every cluster has at least k elements. Computing cluster centroids takes $O(nd)$ initially and $O(kd)$ for each new cluster resulting from a merge operation. Further, given the cluster centroids are known, it takes $O(n/k\ d)$ to compute distances to all centroids for any cluster that needs to be merged. In total, MERGE takes $O(nd + n/k \cdot (kd + n/k\ d))$ which is equal to $O(n^2dk^{-2})$ for $k \leq \sqrt{n}$. To analyze the time complexity of SPLIT, the number of MDAV$^+$ executions and the respecting database sizes have to be considered. As the time complexity of MDAV$^+$ is quadratic in n, the worst case would be a single cluster containing all elements, which is a scenario that can indeed occur after the execution of MERGE. Assuming this worst case, complexity of SPLIT is bounded by $O(n^2dk^{-1})$. All together, again assuming $k \leq \sqrt{n}$, the time complexity of LMaS is dominated by the complexity of k-MEANS++ and is hence equal to $O(n^2dik^{-1})$. □

Algorithm 11: LMaS

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $\mathcal{C} = \{C_1, \ldots, C_{\lfloor n/k \rfloor}\} \leftarrow k$-MEANS++$(X, \lfloor n/k \rfloor)$
$\mathcal{C} \leftarrow$ MERGE$(\mathcal{C}, k)$
$\mathcal{C} \leftarrow$ SPLIT$(\mathcal{C}, k)$

Algorithm 12: MERGE

input : clustering $\mathcal{C}$ and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

repeat
  foreach $C_i \in \mathcal{C}$ **do**
    if $0 < |C_i| < k$ **then**
      Let $C_j \leftarrow \arg\min_{C \in \mathcal{C}} \delta(c(C_i), c(C))^2$
      $C_i \leftarrow C_i \cup C_j$
      $C_j \leftarrow \emptyset$
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C \in \mathcal{C} \mid |C| = 0\}$
until $\neg\exists C \in \mathcal{C}$ *with* $|C| < k$

Correctness of LMaS should be obvious. Although k-MEANS++ is used, its approximation guarantee does not translate to LMaS and the k-anonymous microaggregation problem. This is because in a worst case, all clusters generated by k-MEANS++ have to be merged to obtain k-anonymity.[5] For experimental results of LMaS see section 6.5.

[5] Consider a case in which $\lfloor n/k \rfloor - 1$ clusters of size 1 and one cluster of all remaining elements are created and $\lfloor n/k \rfloor - 1 < k$.

Algorithm 13: SPLIT

input : clustering $\mathcal{C}$ and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

```
Let C' ← ∅
foreach C_i ∈ C do
    if |C_i| ≥ 2k then
        C' ← C' ∪ MDAV+(C_i, k)
    else
        C' ← C' ∪ {C_i}
C ← C'
```

6.4 ONA and ONA*

ONA is another Lloyd-based microaggregation heuristic introduced by Soria-Comas et al. in 2019 [63]. Despite its full name (Near-Optimal microAggregation) no approximation guarantees are known. Nevertheless, ONA is the first microaggregation heuristic able to adapt LLOYD to produce a k-member clustering even for small k on small databases.

ONA starts with a randomly generated k-member clustering and repeats the following alternating steps until convergence: Iterate over all elements x and consider their cluster $C(x)$. If $C(x)$ has more than k elements try to lower information loss by reassigning x to another cluster $C_{\text{target}}(x)$. If $|C(x)| = k$ try to improve the clustering by dissolving $C(x)$ and redistribute its elements $s \in C(x)$ to other clusters $C_{\text{target}}(s)$ nearby. If the cost for dissolving ($costs_D$) is higher than the current cost of keeping the status quo ($cost_K$), nothing is changed. Finally, split all clusters that have grown to size at least $2k$ by applying ONA recursively. This rearrangement of elements is stopped when within a round no change has occurred or a preset number of rounds has been reached.

As can be seen in the pseudocode of ONA given as algorithm 14, the clustering process resembles that of LLOYD. Both start with a random initial clustering and move elements to clusters with closer centroids. The main difference is that ONA does not move an element out of a cluster, if this would violate the k-member condition. Further, ONA might dissolve or create new clusters in a greedy way if there is some distortion to be avoided. Of course, ONA is not able to start with an initial clustering generated by the Forgy method or k-MEANS++, as a valid k-member clustering is required at any given stage of the algorithm. Instead, ONA starts with a random clustering consisting of $\lfloor n/k \rfloor$ clusters with at least k elements each.

Like LLOYD and LMaS, ONA suffers from unreliability caused by the probabilistic initialization with a randomly generated k-member clustering. As for these algorithms, a bad initialization inevitably leads to a bad output. While the authors do not provide any guidelines on how to tackle this problem, the standard approach would be to repeat the algorithm several times and output the clustering with the best solution found, as proposed for LMaS, as well.

The authors claim that ONA has the same worst case time complexity of $O(ndki)$ as

Algorithm 14: ONA [63]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Randomly generate k-member clustering $\mathcal{C} = \{C_1, \ldots, C_{m'}\}$
repeat
  foreach $x \in X$ **do**
    if $|C(x)| > k$ **then**
      // Should x be reassigned to another cluster?
      $C(x) \leftarrow C(x) \setminus \{x\}$
      Let $C_{\text{target}}(x) \leftarrow \arg\min_{C \in \mathcal{C}} \delta(x, c(C))^2$
      $C_{\text{target}}(x) \leftarrow C_{\text{target}}(x) \cup \{x\}$
    else if $|C(x)| = k$ **then**
      // Should cluster $C(x)$ be dissolved?
      foreach $s \in C(x)$ **do**
        Let $C_{\text{target}}(s) \leftarrow \arg\min_{C \in \mathcal{C} \setminus \{C(x)\}} \delta(s, c(C))^2$
      Let $\mathcal{C}_{\text{target}} \leftarrow \bigcup_{s \in C(x)} \{C_{\text{target}}(s)\}$
      Let $\mathcal{C}'_{\text{target}} \leftarrow \emptyset$
      foreach $C \in \mathcal{C}_{\text{target}}$ **do**
        $\mathcal{C}'_{\text{target}} \leftarrow \mathcal{C}'_{\text{target}} \cup \{C \cup \{s \in C(x) \mid C_{\text{target}}(s) = C\}\}$
      Let $cost_K \leftarrow cost(C(x)) + cost(\mathcal{C}_{\text{target}})$
      Let $cost_D \leftarrow cost(\mathcal{C}'_{\text{target}})$
      if $cost_K > cost_D$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \setminus (\{C(x)\} \cup \mathcal{C}_{\text{target}})$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'_{\text{target}}$
  // Split clusters that have become too large
  foreach $C \in \mathcal{C}$ **do**
    if $|C| \geq 2k$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \text{ONA}(C, k)$
until *convergence*

LLOYD but do not provide proof. Accounting for the fact that there are $O(n/k)$ instead of $O(k)$ clusters in ONA and further that the dissolve operation reclusters $O(k)$ elements at once indicate that the time complexity of ONA cannot be linear in n. My analysis summarized in theorem 6.5 and its proof shows that the time complexity of ONA is in fact quadratic in n.

Theorem 6.5.
Assuming ζ is the total number of rounds until convergence and ignoring any complexity of recursive splitting of clusters, the worst case time complexity of algorithm 14 is bounded by $O(\zeta(n^2 + nk^2)d)$.

Proof. To analyze complexity, let us break down the algorithm into several parts.

- **Initialization:** The time used for random initialization can be considered linear in n. Further, initial computation of all centroids takes $O(nd)$.
- **Reassignment:** Reassignment takes time $O(ndk^{-1})$ as there are $O(n/k)$ clusters at any given time and distances in d dimensions need to be computed from x to all cluster centroids. Updating the centroid of the target cluster takes $O(kd)$.
- **Dissolving:** A dissolve process uses $O(nd)$ to find $C_{\text{target}}(s)$ for all $s \in C(x)$. Both, $\mathcal{C}_{\text{target}}$ and $\mathcal{C}'_{\text{target}}$ can be computed in $O(n)$. It takes $O(k^2 d)$ to calculate cost for $cost_K$ and $cost_D$ as well as to update all centroids of affected clusters, when $C(x)$ is indeed dissolved. All together, an execution of the dissolve process takes $O((n + k^2)d)$ regardless of whether the cluster is actually dissolved or not.
- **Splitting:** After an element x is handled, clusters that grew to $2k$ or more elements are split. For this, ONA is applied recursively on databases of size $O(k)$. As it is not trivial to bound the maximum number of splits necessary per execution of the outer loop of algorithm 14, it is not clear whether splitting increases the asymptotic complexity of ONA as a whole. However, without knowing the complexity of ONA including splits, we cannot bound the time complexity of a single split.

Putting things together and further ignoring additional complexity for splitting clusters, an execution of the outer loop beginning in step 2 of algorithm 14 takes time $O((n^2 + nk^2)d)$. Including the number of repetitions ζ until convergence, the total time complexity of ONA is $O(\zeta(n^2 + nk^2)d)$. □

Regarding information loss ONA seems to be comparable to previous quadratic time heuristics on real and synthetic benchmark databases. However, it was not possible for me to reproduce the excellent empirical results claimed in [63]. Considering its problems and relatively small improvements in regard to information loss of ONA compared to LMaS and MDAV variants in the experiments shown in section 6.5, the question arises whether a native Lloyd-based microaggregation heuristic can indeed outperform traditional heuristics significantly.

ONA*

The algorithm ONA* published in cooperation with Rüdiger Reischuk in 2021 [71] confirms this hypothesis. Improving upon ONA in several aspects, ONA* is able to anonymize databases with consistently lower information loss than MDAV^*_γ, LMaS, or ONA while being deterministic.

Replacing the random initial clustering by a good deterministic process increases the performance significantly. We have analyzed the methodology of generating and rearranging clusters in detail and designed better selection strategies. More precisely, we have investigated how the strategy of rearranging clusters in ONA can be improved. It turned out that iterating over data points in an arbitrary order, which seems to be a good strategy for k-clustering, is not as good for k-member clustering. Instead, iterating over clusters in a well chosen order is computationally more efficient and, more importantly, gives better utility. To do this, ONA* makes a more precise estimation concerning reassignment. Whereas ONA bases its decision, whether and where to move an element x, solely on the distances to centroids, ONA* also compares the actual cost before and after a possible reassignment. The result is used to decide which element from a cluster is reassigned first and, of course, whether any element should be reassigned at all. A final modification simplifies matters substantially. Every cluster to be split by ONA has between $2k$ and $3k-1$ elements and should consequently be divided into two parts. For this task ONA is not likely to find better solutions than MDAV algorithms, but requires more time and risks computing a bad clustering due to its probabilistic initialization. Hence, we have replaced the recursive execution of ONA by a call to MDAV*, which first creates 2 clusters with exactly k elements on opposite sides of the global centroid and afterwards assigns remaining elements to their closest cluster which is likely to yield an optimal solution in this special case. A complete description of ONA* is given as algorithms 15 to 17. Experiments comparing information loss and runtime to variants of MDAV and other Lloyd-based heuristics are given in section 6.5.

Algorithm 15: ONA*

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

```
1  Let C = {C_1, ..., C_m'} ← MDAV*(X, k)
   // Split clusters that are too large
2  foreach C ∈ C do
3      if |C| ≥ 2k then
4          C ← C \ {C}
5          C ← C ∪ MDAV+(C, k)
6  repeat
       // Phase 1: dissolving clusters
7      foreach C_i ∈ C with |C_i| = k do
8          C ← ONA*-Dissolve(C, C_i)
       // Phase 2: reassigning elements
9      foreach C_i ∈ C with |C_i| > k do
10         C ← ONA*-Reassign(C, C_i)
11 until convergence
```

Correctness of ONA* is given by the fact that it is initialized with a valid k-member clustering generated by MDAV*and that neither split, nor dissolve, nor reassign are able to create clusters with less than k elements. Note that the initial splitting loop is beneficial

Algorithm 16: ONA*-Dissolve

input : k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$ and cluster C_i
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_{m'}\}$

foreach $s \in C_i$ **do**
  Let $C_{\text{target}}(s) \leftarrow \arg\min_{C \in \mathcal{C} \setminus \{C_i\}} \delta(s, c(C))^2$
Let $\mathcal{C}_{\text{target}} \leftarrow \bigcup_{s \in C_i} \{C_{\text{target}}(s)\}$
Let $\mathcal{C}'_{\text{target}} \leftarrow \emptyset$
foreach $C \in \mathcal{C}_{\text{target}}$ **do**
  $\mathcal{C}'_{\text{target}} \leftarrow \mathcal{C}'_{\text{target}} \cup \{C \cup \{s \in C_i \mid C_{\text{target}}(s) = C\}\}$
Let $cost_K \leftarrow cost(C_i) + cost(\mathcal{C}_{\text{target}})$
Let $cost_D \leftarrow cost(\mathcal{C}'_{\text{target}})$
if $cost_K > cost_D$ **then**
  $\mathcal{C} \leftarrow \mathcal{C} \setminus (\{C_i\} \cup \mathcal{C}_{\text{target}})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'_{\text{target}}$
  // Split clusters that have become too large
  foreach $C \in \mathcal{C}'_{\text{target}}$ **do**
    if $|C| \geq 2k$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \text{MDAV}^*(C, k)$

Algorithm 17: ONA*-Reassign

input : k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$ and cluster C_i
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_{m'}\}$

repeat
  foreach $s \in C_i$ **do**
    Let $C_{\text{target}}(s) \leftarrow \arg\min_{C \in \mathcal{C} \setminus \{C_i\}} \delta(s, c(C))^2$
    Let $improvement(s) \leftarrow$
     $(cost(C_i) - cost(C_i \setminus \{s\})) - (cost(C_{\text{target}}(s) \cup \{s\}) - cost(C_{\text{target}}(s)))$
  Let $s' = \arg\max_{s \in C_i} improvement(s)$
  if $improvement(s') \leq 0$ **then**
    break
  else
    $C_i = C_i \setminus \{s'\}$
    $C_{\text{target}}(s') \leftarrow C_{\text{target}}(s') \cup \{s'\}$
    // Split clusters that have become too large
    if $|C_{target}(s')| \geq 2k$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_{\text{target}}(s')\}$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \text{MDAV}^*(C_{\text{target}}(s'), k)$
until $|C_i| = k$

because MDAV* does not guarantee a maximum cluster size of $2k-1$ in general. However, as shown in fact 6.6 when used as a split-subroutine on databases of size at most $3k-1$, MDAV* guarantees a maximal cluster size of $2k-1$ in the resulting clustering.

Fact 6.6.
Let $\mathcal{C} = \texttt{MDAV}^*(X, k)$ for $|X| \leq 3k-1$ and $k \in \mathbb{N}$.

$$\forall C \in \mathcal{C} : \ |C| \leq 2k-1$$

Proof. MDAV* starts by creating a cluster C_1 with k elements. Now there are at most $2k-1$ elements remaining. As MDAV* always creates a new cluster, when there are k elements remaining, C_1 cannot be extended by k or more elements. Hence, C_1 can never grow to $2k$ or more elements. However, if a second cluster C_2 is created at any time throughout the run of the algorithm, at most $k-1$ elements are left to be assigned to C_1 or C_2, so also C_2 cannot obtain more than $2k-1$ elements. □

For a low information loss, applicability of fact 6.6 and validity of the upcoming complexity analysis of ONA* it is required that any cluster $C \in \mathcal{C}$ has less than $2k$ elements before entering the main loop. However, as noted before, MDAV* is not able to give this guarantee when applied to large sets of elements. Although these situations should occur very rarely and never happened during my experiments, MDAV$^+$ instead of MDAV* is applied in initial splits. Unlike MDAV*, MDAV$^+$ can by design never create a cluster of $2k$ or more elements. Nevertheless, MDAV$^+$is not used in other parts of ONA* as MDAV* is able to deliver lower information loss in general.

Theorem 6.7.
Assuming ζ is the total number of iterations of ONA*, *the worst case time complexity of algorithm 15 is bounded by* $O((n^2 + \zeta\ (n^2 + n\ k^2))\ d)$.

Proof. To determine the time complexity of ONA* let us again consider its basic building blocks and let ζ be the number of repetitions until convergence. There are at most n/k clusters C_i of size k which might be dissolved in phase 1. For each element s of such a cluster its closest cluster $C_{\text{target}}(s)$ can be found in time $O(n/k\ d)$. For each cluster C_i evaluating the cost function for it and the at most k clusters in $\mathcal{C}_{\text{target}}$ takes time $O(k^2 d)$.

Splitting a cluster C_ℓ by MDAV* requires $O(|C_\ell|^2\ d)$ time. Each cluster C_i can give rise to at most k splits of a cluster C_ℓ of size less than $3k$, which adds up to $O(k^3 d)$ computational effort. Thus the total time of Phase 1 can be bounded by

$$n/k \cdot (k \cdot O(n/k\ d) + O(k^2 d + k^3 d)) = O((n^2/k + n\ k^2)\ d).$$

For Phase 2 one has to consider at most n/k clusters of size between $k+1$ and $2k-1$. The loop starting in step 1 of algorithm 17 is executed less than k times. In each execution, the computation of the closest centroid and the improvement for each element s of a cluster take time $O(n/k\ d)$ and $O(k\ d)$ respectively. Now there can be at most one split adding time $O(k^2\ d)$. Hence, per cluster C_i at most $O(n\ k\ d) + O(k^3\ d)$ time is needed. All together this gives the upper bound

$$n/k \cdot (O(n\ k\ d) + O(k^3\ d)) = O((n^2 + n\ k^2)\ d)$$

for Phase 2. If the time $O(n^2\ d)$ for the initialization by MDAV*, initial splits and initial calculation of cluster centroids is added we finally get

$$O((n^2 + \zeta\ (n^2 + n\ k^2))\ d).$$

□

Beyond ONA*

There are, of course, still some ways of improvement for ONA*. One of those improvements would be to initialize ONA* with a heuristic even better than MDAV*. By swapping out MDAV* by MDAV^*_γ, such an improvement could be achieved. As can be seen in table 6.2 in section 6.5 this variant of ONA*, called ONA^*_γ, is indeed able to lower information loss slightly. However, as for MDAV^*_γ the implementation of a gain factor γ has a downside as well. While the runtime for fixed γ is similar to that of ONA*, finding a good γ for a database is time-consuming in itself, as no good strategy besides exhaustive search is known. Nevertheless, when time consumption is secondary, MDAV^*_γ is a valid alternative, especially because its asymptotic time consumption is not different from ONA*.

As ONA*-like re-clustering rounds can be applied to any initial clustering, one might argue that any microaggregation heuristic might be improved by it, as ONA* cannot increase information loss compared to its initial clustering. This approach is the source of the algorithm MONA presented in the next chapter. By acting as a two-phase algorithm, MONA is able to give its users control over a trade-off between time consumption and quality of resulting data.

Overview

Table 6.1: Key properties of all Lloyd-based microaggregation heuristics presented in this chapter.

	Time	Space	Behavior	Introduced	Reference
PCL	$\geq O(n^2)$	$O(nd)$	probabilistic	2013	[58]
LMaS	$O(n^2dik^{-1})$	$O(nd)$	probabilistic	-	-
ONA	$\geq O(\zeta(n^2 + nk^2)d)$	$O(nd)$	probabilistic	2019	[63]
ONA*	$O(n^2 + \zeta(n^2 + nk^2))d)$	$O(nd)$	deterministic	2021	[71]
ONA^*_γ	$O(n^2 + \zeta(n^2 + nk^2))d)$	$O(nd)$	deterministic	-	-

6.5 Experimental Evaluation of Lloyd-based Heuristics

In this section the Lloyd-based heuristics are compared to each other and to a selection of heuristics using the maximum distance principle. The same benchmark databases and general test setting as presented in section 5.2 are used. Further, the probabilistic algorithms LMaS and ONA are indexed with numbers $\mu \in \{1, 10, 100, 1000\}$ in the upcoming tables. An

index of μ indicates that the algorithm has been executed μ times and only the result with lowest information loss found is reported. For tables comparing runtimes, the numbers reported are to be interpreted as the amount of time necessary to perform μ executions. The gain factor γ for $\texttt{ONA}^*_\gamma$ is used for initialization only and is optimized to minimize the information loss of the final result. Hence, gain factors for $\texttt{ONA}^*_\gamma$ might differ from those of $\texttt{MDAV}^*_\gamma$ even if both use γ just to initialize $\texttt{MDAV}^*_\gamma$.

Convergence for LLOYD and k-MEANS++ is assumed when the difference between cost of the previous and current clusterings falls below 0.001 or when 50 rounds have passed, whichever occurs first. Convergence for ONA is assumed when the clustering does not change during a single clustering round. Alternatively, the algorithm is halted when 30 rounds have passed. For nested ONA executions during split operations, the round limit is lowered to 10 on the second level and to 0 on the third level. Hence, nested ONA executions are not allowed to split. Lastly, the convergence of $\texttt{ONA}^*$ and $\texttt{ONA}^*_\gamma$ is handled like in ONA, without the need to further limit split operations, as these are handled by $\texttt{MDAV}^*$ instead of ONA. In table 6.5 the number of rounds until convergence is shown for ONA, $\texttt{ONA}^*$ and $\texttt{ONA}^*_\gamma$ on Census for different values of k. It can be observed that convergence is usually reached in less than 10 rounds.

In figure 6.4 the consistency of 1000 runs of LMaS and ONA is displayed and compared on the Census database for $k = 10$. Besides the obvious fact, that almost all executions of ONA result in a lower information loss than even the best run of LMaS, the difference in consistency is also of interest. It can be observed that results of ONA runs are more predictable and stable. Hence, less executions of ONA are needed to obtain reliable results. Both heuristics have in common, that by increasing the number of repetitions μ the probability of finding a very bad solution grows faster than the probability of finding a very good one. It seems as if the average solution is not far from the best one achievable in a reasonable amount of time. Hence, few runs are needed to find relatively good solutions. Table 6.3 further indicates that for both heuristics increasing the number of repetitions μ increases the highest information loss observed by a lot, while lowest information loss and mean change only slightly. As can be seen, there is some improvement to be gained by increasing μ. But when a good confidence is aimed at, this increases the runtime significantly.

The general picture of information loss over all benchmark databases and values of k according to section 5.2 is drawn in table 6.2. While LMaS reaches results comparable with those of $\texttt{MDAV}^*_\gamma$, ONA is able to lower information losses below those achievable by any MDAV variant or LMaS. Finally, $\texttt{ONA}^*$ is shown to be the algorithm with the lowest information loss over all previous algorithms. It is topped only by $\texttt{ONA}^*_\gamma$ which uses the extended cluster extension mechanism of $\texttt{MDAV}^*_\gamma$ and a gain factor γ in its initialization.

If we include the time consumption (see table 6.6, 6.7, 6.8 and appendix B) in our analysis, a different conclusion needs to be made. As single executions of LMaS and ONA are not able to deliver reliable results, they are needed to be executed several times, increasing runtime significantly. Furthermore, even single executions of ONA require significantly more time than any MDAV variant. Especially for large databases like Credit Card, execution times of LMaS and ONA outweigh any improvements made over $\texttt{MDAV}^*$ or $\texttt{MDAV}^*_\gamma$. Even including runtime, $\texttt{ONA}^*$ seems to be the best candidate here. It delivers lower information loss than ONA or LMaS even if the probabilistic heuristics are allowed to be executed 100 times, while $\texttt{ONA}^*$ needs to be run only once. Also there is no need to find a good gain factor γ as for

$\texttt{MDAV}^*_\gamma$, as $\texttt{ONA}^*$ does not need to be adapted for different databases. As $\texttt{ONA}^*$ takes about twice as long as $\texttt{MDAV}^*_\gamma$ regardless of database size or k, it clearly shows that Lloyd-based microaggregation heuristics are indeed superior to maximum distance based heuristics.

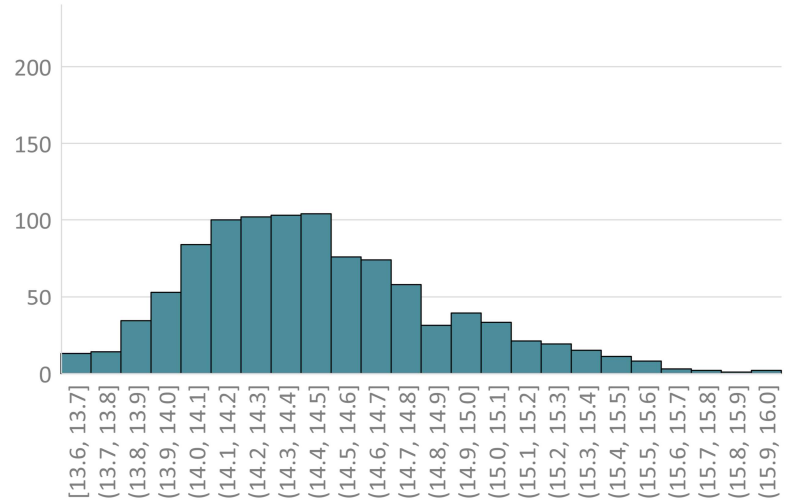
One side note to consider is an anomaly regarding the runtime of ONA. Unlike any other microaggregation heuristic tested so far, the time consumption of ONA does not decrease consistently with increasing k. This is despite the fact that the asymptotic worst case time complexity of ONA is not very different from that of e.g. $\texttt{ONA}^*$. A possible explanation is that $\texttt{ONA}^*$ relies heavily on the execution of $\texttt{MDAV}^*$, whose average time complexity is anti-proportionally dependent on k.

Table 6.2: Average percentage information loss difference between a selection of maximum distance and Lloyd-based heuristics over all benchmarks. A value below 100 indicates that an algorithm causes less information loss on average. Differences based on $\texttt{ONA}_{100}$ do not include data from the Adult and Credit Card databases, as they are too big to be handled with $\texttt{ONA}_{100}$ within a reasonable amount of time.

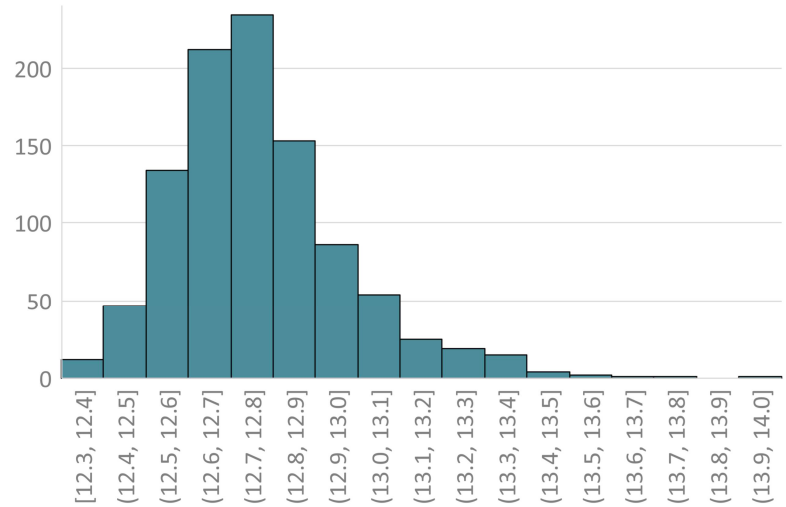
	Information Loss Differences in %						
from\to	$\texttt{MDAV}^+$	$\texttt{MDAV}^*$	$\texttt{MDAV}^*_\gamma$	$\texttt{LMaS}_{100}$	$\texttt{ONA}_{100}$	$\texttt{ONA}^*$	$\texttt{ONA}^*_\gamma$
$\texttt{MDAV}^+$	100	92.7	89.9	90.1	86.3	83.0	82.1
$\texttt{MDAV}^*$		100	96.7	96.7	92.5	89.2	88.2
$\texttt{MDAV}^*_\gamma$			100	99.9	95.9	92.2	91.2
$\texttt{LMaS}_{100}$				100	96.1	92.3	91.3
$\texttt{ONA}_{100}$					100	96.6	95.4
$\texttt{ONA}^*$						100	98.9
$\texttt{ONA}^*_\gamma$							100

Table 6.3: Statistics of the consistency of outputs on different numbers of LMaS and ONA executions on the Census benchmark database for $k = 10$. The runtime and information losses of $\texttt{MDAV}^*$ and $\texttt{ONA}^*$ are included for reference. As $\texttt{MDAV}^*$ and $\texttt{ONA}^*$ are deterministic, they are executed only once.

	lowest IL	highest IL	median	mean	variance	runtime
$\texttt{MDAV}^*$	14.00	14.00	14.00	14.00	0.00	$0s$
$\texttt{LMaS}_{10}$	13.58	14.73	14.12	14.16	0.11	$0s$
$\texttt{LMaS}_{100}$	13.56	15.47	14.28	14.38	0.21	$1s$
$\texttt{LMaS}_{1000}$	13.56	15.95	14.36	14.41	0.17	$10s$
$\texttt{ONA}_{10}$	12.50	13.11	12.83	12.84	0.04	$1s$
$\texttt{ONA}_{100}$	12.49	13.51	12.77	12.80	0.04	$7s$
$\texttt{ONA}_{1000}$	12.33	13.99	12.77	12.80	0.04	$67s$
$\texttt{ONA}^*$	12.46	12.46	12.46	12.46	0.00	$0s$



(a) LMaS



(b) ONA

Figure 6.4: Histograms of the information losses of 1000 executions of LMaS and ONA on the Census database for $k = 10$. Both histograms use a bin size of 0.1. It can be observed that ONA is more consistent. As both histograms are left-leaning (towards low information loss), few runs are needed to find relatively good solutions.

Table 6.5: Number of clustering rounds ζ until convergence for ONA and ONA* on the Census database for different values of k. As ONA* and ONA^*_γ are deterministic, they are executed only once.

	k	min ζ	mean ζ	max ζ
ONA_{1000}	2	4	5.7	11
ONA_{1000}	5	5	8.9	17
ONA_{1000}	10	7	11.8	30
ONA*	2		5	
ONA*	5		3	
ONA*	10		6	
ONA^*_γ	2		3	
ONA^*_γ	5		4	
ONA^*_γ	10		7	

Table 6.6: Information losses and runtimes of a selection of maximum distance and Lloyd-based heuristics on EIA $\in \mathcal{X}_{4092,11}$.

	Information Loss in % on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MDAV^+	0.32	0.49	0.67	1.78	2.21	3.55
MDAV*	0.22	0.45	0.62	0.91	2.03	2.63
MDAV^*_γ	0.20	0.39	0.54	0.82	1.66	2.18
LMaS_{100}	0.19	0.39	0.56	0.80	1.67	2.04
ONA_{100}	0.21	0.40	0.59	0.80	1.60	2.01
ONA*	0.20	0.37	0.52	0.79	1.63	1.99
ONA^*_γ	0.19	0.37	0.52	0.78	1.58	1.98

	Runtime in s on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MDAV^+	0.11	0.06	0.05	0.04	0.03	0.02
MDAV*	0.37	0.24	0.18	0.14	0.10	0.09
MDAV^*_γ	0.30	0.22	0.18	0.18	0.18	0.14
LMaS_{100}	24.86	21.95	19.06	18.24	15.14	11.94
ONA_{100}	78.41	97.59	56.13	147.08	24.84	21.68
ONA*	0.60	0.40	0.29	0.27	0.21	0.17
ONA^*_γ	0.51	0.38	0.30	0.28	0.32	0.21

	γ used on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
ONA^*_γ	1.6	1.0	0.9	0.5	1.8	1.3

Table 6.7: Information losses and runtimes of a selection of maximum distance and Lloyd-based heuristics on Credit Card $\in \mathcal{X}_{30000,24}$. ONA is executed only 10 times in this tests, as 100 executions are too time consuming.

	Information Loss in % on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MDAV$^+$	3.65	6.39	8.49	10.17	12.59	15.21
MDAV*	3.65	6.44	8.48	10.22	12.36	14.67
MDAV$^*_\gamma$	3.59	6.25	8.25	9.83	12.11	14.47
LMaS$_{100}$	3.89	6.48	7.92	9.28	11.36	13.42
ONA$_{10}$	3.98	6.16	7.58	8.72	10.53	12.30
ONA*	3.50	5.86	7.53	8.65	10.23	12.24
ONA$^*_\gamma$	3.50	5.86	7.40	8.46	10.20	12.12

	Runtime in s on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MDAV$^+$	6.7	4.6	3.3	2.7	2.0	1.5
MDAV*	24.5	17.2	14.4	12.7	10.8	9.2
MDAV$^*_\gamma$	28.2	19.3	16.0	13.7	12.1	9.8
LMaS$_{100}$	5438.9	4670.9	4138.9	3527.3	2821.5	2207.2
ONA$_{10}$	961.9	1333.4	1188.5	1041.8	958.8	869.7
ONA*	49.3	42.9	35.4	36.5	32.8	29.6
ONA$^*_\gamma$	52.1	58.9	52.4	41.9	35.5	25.2

	γ used on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
ONA$^*_\gamma$	1.1	1.0	1.7	1.2	1.1	0.9

Table 6.8: Mean information losses and runtimes of a selection of maximum distance and Lloyd-based heuristics on sets of synthetic databases SimC = $\{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$.

	Mean Information Loss in % on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MDAV$^+$	3.82	6.86	9.23	11.04	14.11	18.09
MDAV*	3.36	6.00	8.17	9.66	12.32	15.69
MDAV$^*_\gamma$	3.28	5.73	7.61	9.16	11.65	14.84
LMaS$_{100}$	3.45	5.44	6.87	8.08	10.06	12.57
ONA$_{100}$	3.54	5.40	6.69	7.72	9.36	11.54
ONA*	3.16	5.20	6.59	7.68	9.33	11.40
ONA$^*_\gamma$	3.14	5.15	6.51	7.55	9.20	11.24

	Mean Runtime in s on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MDAV$^+$	0.01	0.01	0.01	0.01	0.00	0.00
MDAV*	0.05	0.03	0.03	0.02	0.02	0.02
MDAV$^*_\gamma$	0.05	0.03	0.03	0.03	0.03	0.02
LMaS$_{100}$	2.55	1.67	1.22	1.11	0.96	0.83
ONA$_{100}$	13.31	11.72	10.15	9.27	7.74	6.23
ONA*	0.09	0.07	0.06	0.05	0.05	0.05
ONA$^*_\gamma$	0.08	0.07	0.07	0.06	0.06	0.05

	Mean γ used on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
ONA$^*_\gamma$	1.1	1.2	1.0	1.0	0.9	0.7

7

Heuristics for Large Databases

In the previous two chapters we concentrated on heuristics that are able to introduce less and less information loss while operating in the same rough time complexity of $O(n^2)$. Unfortunately, these heuristics are not fast enough to handle very large databases of millions of elements. In this chapter, we present solutions to this problem. In the first section an efficient $O(nd \log n)$ k-member clustering algorithm is adapted to deliver acceptable results for microaggregation. Improving this, section 7.2 focuses on combining the new heuristic with ONA* to obtain a scalable class of heuristics MONA, which is able to operate between near linear and quadratic time complexity while delivering adequate information losses for each configuration. In section 7.3 an experimental evaluation is made.

7.1 Using MONDRIAN for Microaggregation

In 2006 LeFevre et al. introduced MONDRIAN, an anonymization algorithm that achieves k-anonymity in $O(nd \log n)$ time [37]. The optimization goal of MONDRIAN is to create clusters with cluster sizes as close to k as possible. Unlike previously presented heuristics, MONDRIAN has not been designed for the k-anonymous microaggregation problem. Instead, MONDRIAN is a local recoding scheme, aiming at minimizing the discernability metric C_{DM} and the normalized average equivalence class size metric C_{AVG} of databases potentially including numerical and categorical data as defined in section 3.2.

MONDRIAN may not be defined as a microaggregation algorithm, but it creates a k-member clustering in the process of anonymization. Hence, its strategy can be used to perform k-anonymous microaggregation by calculating and reporting centroids for each cluster created. However, the question arises whether the resulting information loss is comparable to state-of-the-art heuristics designed to minimize information loss.

The lower time complexity of MONDRIAN compared to maximum distance or Lloyd-based heuristics is caused by the fact that no distances between elements are computed. Instead, MONDRIAN resembles the process of sub-dividing a d-dimensional space by d-dimensional trees. A database is interpreted as a d-dimensional space with the elements being points in that space. In the first step, MONDRIAN splits the database into two clusters by projecting it onto one of its d dimensions and dividing elements at the median. Subsequently, clusters are divided further, potentially using different *splitting dimensions* for different (sub)clusters. A cluster is no longer split and considered final if a split at the median would result in at

least one new cluster having less than k elements. Thus, in the final clustering the size of each cluster is between k and $2k - 1$.

Choosing a good splitting dimension for each cluster is a crucial part of the algorithm when used as a microaggregation heuristic. Especially for higher-dimensional data, choosing a less optimal splitting dimension might result in big and sparsely populated clusters, resulting in high information loss. But, as MONDRIAN is designed to minimize C_{DM} or C_{AVG}, the choice of splitting dimensions has been far less important. MONDRIAN chooses the splitting dimension for any cluster as the attribute dimension with widest range of values in that cluster, a strategy aimed at reducing the area of clusters as far as possible. A pseudocode of MONDRIAN is given as algorithm 18.

Algorithm 18: MONDRIAN [37]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

if $|X| < 2k$ **then**
  return X
Let $dim \leftarrow \arg\max_{j \in \{1,\ldots,d\}} \left(\max_{x_i \in X} x_i^j - \min_{x_i \in X} x_i^j \right)$
Let $median \leftarrow median(\{x_i^{dim} \mid x_i \in X\})$
Let $lhs \leftarrow \emptyset$
Let $rhs \leftarrow \emptyset$
foreach $x_i \in X$ **do**
  if $x_i^{dim} \leq median$ **then**
    $lhs \leftarrow lhs \cup \{x_i\}$
  else
    $rhs \leftarrow rhs \cup \{x_i\}$
return MONDRIAN$(lhs, k) \cup$ MONDRIAN(rhs, k)

As can be seen in section 7.3, MONDRIAN is not able to create clusterings with information losses as low as those of ONA or MDAV variants. However, as expected it takes significantly less time. The clustering strategy of MONDRIAN can be interpreted as acting in rounds of splitting every existing cluster of size at least $2k$ into two smaller clusters. There are $O(\log n)$ splitting rounds where every element is assigned to a new, smaller cluster. Computation of the splitting dimension can be done in time linear in d and n and computation of the median can be done in $O(n)$. Hence, the total time complexity of MONDRIAN is $O(nd \log n)$.

Choosing splitting dimensions according to the *widest range* rule might be problematic as information loss is defined by cluster density rather than cluster area. During the work for publication [71] I have investigated several alternative splitting criteria with the same asymptotic time complexity and reached the conclusion that a significant improvement could be achieved by choosing the splitting dimension as the dimension with the largest variance of values. The resulting algorithm called MONDRIAN_V achieves about 15% lower information loss on average over MONDRIAN in the tests provided in section 7.3. The splitting

rule of MONDRIAN_V has the same time complexity of $O(nd)$ and can be formalized as

$$dim \leftarrow \arg\max_{j\in\{1,\dots,d\}} \left(\sum_{x_i \in X} \left(x_i^j - c(X)^j \right)^2 \right).$$

A pseudocode of MONDRIAN_V is not included, as it is the same as algorithm 18 with step 3 replaced by the new splitting rule.

The improvement going from MONDRIAN to MONDRIAN_V shows that even for low-dimensional data, the choice of the right way to split is quite important. A natural next step is to increase the number of options for splits. Up to this point only splits according to attribute values in a single dimension have been considered. The largest possible set of splits would be the set of all hyperplanes dividing the database into two parts with a varying amount of elements on each side. However, deciding which splitting hyperplane to choose is a time consuming process, eliminating the performance gains made by MONDRIAN_V over ONA*.

In MONDRIAN_V splits can be interpreted as hyperplanes perpendicular to one of the unit vectors $e_1, \dots, e_d$ of the data space $\mathbb{R}^d$ dividing the elements into two clusters. The next MONDRIAN based algorithm presented in [71] called MONDRIAN_V2D considers additional splits. We now also allow hyperplanes that are perpendicular to a combination $e_{j_1 j_2}$ of a pair of unit vectors $e^+_{j_1 j_2} = \frac{1}{\sqrt{2}} \cdot (e_{j_1} + e_{j_2})$ and $e^-_{j_1 j_2} = \frac{1}{\sqrt{2}} \cdot (e_{j_1} - e_{j_2})$. In other words, the set of possible splits is expanded by hyperplanes that are 45° or 315° between any two unit vectors, see Figure 7.1. As before, splits are made at the median of the dimension (or combination of dimensions) with largest variance. Note that, by the prefactor $\frac{1}{\sqrt{2}}$ we ensure measuring variances in an orthonormal basis resulting in values comparable to those measured along original dimensions which remains a valid option considered by MONDRIAN_V2D.

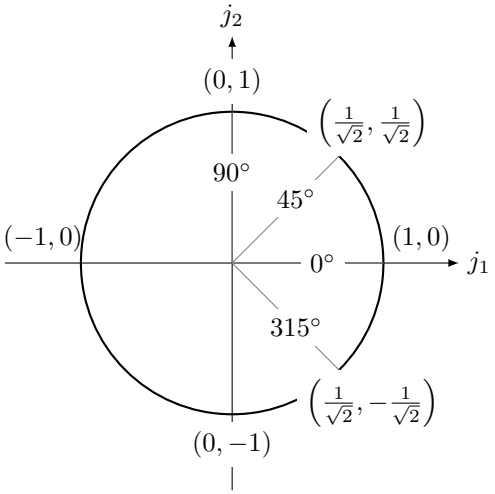


Figure 7.1: Splits in MONDRIAN_V2D are perpendicular to a dimension j or to a combination of two dimensions j_1 and j_2.

The number of possible splits for any given cluster increases from d to $2 \cdot \binom{d}{2} + d = d^2$ since there are $\binom{d}{2}$ pairs of dimensions to choose from and two orientations for each pair together with the d options to split along a single dimension as before. The time complexity

of MONDRIAN_V2D increases to $O(nd^2 \log n)$, but information loss further decreases as shown in section 7.3. A pseudocode for MONDRIAN_V2D is given as Algorithm 19.

Algorithm 19: MONDRIAN_V2D

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

if $|X| < 2k$ **then**
  return X
Let $(dim1, dim2, orientation) \leftarrow$
  $\arg\max_{j_1, j_2 \in \{1,\ldots,d\}, o \in \{-1,1\}} \left(\sum_{x_i \in X} \left(\frac{1}{\sqrt{2}} \cdot \left(x_i^{j_1} + o \cdot x_i^{j_2} \right) \right) \right)$
Let $median \leftarrow median(\{x_i^{dim1} + orientation \cdot x_i^{dim2} \mid x_i \in X\})$
Let $lhs \leftarrow \emptyset$
Let $rhs \leftarrow \emptyset$
foreach $x_i \in X$ **do**
  if $x_i^{dim} \leq median$ **then**
    $lhs \leftarrow lhs \cup \{x_i\}$
  else
    $rhs \leftarrow rhs \cup \{x_i\}$
return MONDRIAN_V2D$(lhs, k) \cup$ MONDRIAN_V2D(rhs, k)

Of course, one could go even further and take combinations of 3 or more unit vectors and additional angles between dimensions increasing the number of choices significantly. When choosing 3 dimensions, there are $\binom{d}{3}$ combinations plus additional possibilities of vector directions within the spanned hypersphere. While this would most probably decrease information loss further, time complexity starts to become an issue. While for low-dimensional data the prospective reduction in information loss is limited, for high-dimensional data, time consumption gets close to or even exceeds that of efficient heuristics using distance computations like MDAV$^+$ delivering significantly lower information losses. See e.g. table 7.6 for the effect on time consumption even two-dimensional splitting might have.

7.2 Combining ONA* and MONDRIAN_V

Computing distances between elements is an important part of clustering small sets of elements in homogeneous clusters. However, as databases get bigger, individual distances between elements are less and less important. Even a weak indicator like variance, separating elements in great distance from each other, is sufficient to do a rough clustering. Hence, the main use of MONDRIAN_V and MONDRIAN_V2D for microaggregation should be to avoid bad decisions like splitting through a natural cluster while creating chunks of data that are small enough to be handled by better but slower algorithms.

In [71] Rüdiger Reischuk and I propose to combine two methods, a MONDRIAN variant as the fast heuristic in the beginning to split large clusters, and then ONA* as the heuristic

of better quality for a fine grained clustering of small clusters. The resulting algorithm is named MONA (Mondrian ONA). The combination is flexibly governed by a parameter ρ that can be chosen between 0 and 1. It defines the switch from MONDRIAN_V to ONA*: clusters of size larger than n^ρ are iteratively split by MONDRIAN_V, smaller ones are then handled by ONA*. Thus, we get a family of algorithms $\texttt{MONA}_\rho$, where $\texttt{MONA}_0$ equals MONDRIAN_V and $\texttt{MONA}_1$ is identical to ONA*. Analogously the algorithm $\texttt{MONA\_2D}_\rho$ combines MONDRIAN_V2D and ONA*. The pseudocode of $\texttt{MONA}_\rho$ is described in algorithm 20. A detailed description of MONA_2D is omitted as it differs only in its splitting rule.

Algorithm 20: MONA (MONDRIAN_V combined with ONA*)

input : database X, minimal cluster size k and split limit n^ρ
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

if $|X| < n^\rho$ **then**
  return $\texttt{ONA}^*(X, k)$
Let $dim \leftarrow \arg\max_{j \in \{1,\ldots,d\}} \left(\sum_{x_i \in X} \left(x_i^j - c(X)^j \right)^2 \right)$
Let $median \leftarrow median(\{x_i^{dim} \mid x_i \in X\})$
Let $lhs \leftarrow \emptyset$
Let $rhs \leftarrow \emptyset$
foreach $x_i \in X$ **do**
  if $x_i^{dim} \leq median$ **then**
    $lhs \leftarrow lhs \cup \{x_i\}$
  else
    $rhs \leftarrow rhs \cup \{x_i\}$
return $\texttt{MONA}(lhs, k, n^\rho) \cup \texttt{MONA}(rhs, k, n^\rho)$

Since ONA* has quadratic time complexity in n, but is only applied to a bunch of smaller databases, the total time complexity in the ONA*-phase is reduced. Furthermore, most computation time of MDAV or ONA variants is due to distance calculations between far apart elements. But this has little influence on the local arrangements of elements. Thus, saving these estimations in the MONDRIAN_V-phase does not increase the information loss much. Still, there might occur a decrease of data quality in the MONDRIAN_V-phase if ONA* would have clustered elements together that lie on both sides of the median of a splitting dimension used by MONDRIAN_V and are now assigned to different sub-problems. However, for larger databases such cases can be expected to have only small influence.

Theorem 7.1.
For $0 < \rho \leq 1$ and $k^2 \leq n^\rho$ the time complexity of $\texttt{MONA}_\rho$ is bounded by $O(n^{1+\rho}\, d)$. When further $d \log n < n^\rho$, this bound holds for $\texttt{MONA\_2D}_\rho$ as well.

Proof. In the ONA*-phase $\texttt{MONA}_\rho$ has to manage $O(\frac{n}{n^\rho}) = O(n^{1-\rho})$ instances with input size at most n^ρ. By applying theorem 6.7 we obtain a time complexity bound of $O((n^{2\rho} + \zeta(n^{2\rho} + n^\rho k^2))d)$ for each of these instances. The time complexity of the MONDRIAN_V-phase is obviously not larger than a complete run of this algorithm. Putting things together, the

total time complexity of $\texttt{MONA}_\rho$ can be bounded by

$$\begin{aligned} &O(nd\log n) + O\left(n^{1-\rho}\right)\cdot O((n^{2\rho}+\zeta(n^{2\rho}+n^\rho k^2))d) \\ &\quad = O(nd\log n) + O((n^{1+\rho}+\zeta(n^{1+\rho}+nk^2))d). \end{aligned}$$

For $\rho > 0$ the first term is majorized by the second. If k is small compared to n, which for larger databases typically holds, and ζ is considered as a constant we get the desired result.

For $\texttt{MONA\_2D}_\rho$ the first term gets an additional factor d. Using the additional requirement $d\log n < n^\rho$ the same bound holds. □

As stated in the introduction of this section, MONDRIAN_V preprocessing can be applied to all microaggregation algorithms. ONA* has been chosen because of its excellent ratio of information loss to time consumption. Nevertheless, one might ask, whether it would be beneficial to use faster algorithms like $\texttt{MDAV}^+$, or adjustable algorithms like $\texttt{ONA}^*_\gamma$. Regarding $\texttt{MDAV}^+$, the results of a small sample of test cases are as expected. For the same values of ρ, the results have slightly higher information loss but consume slightly less time. Although a full test seems necessary to confirm this claim, combining MONDRIAN_V with $\texttt{MDAV}^+$ might be a viable alternative to MONA. Combining MONDRIAN_V with $\texttt{ONA}^*_\gamma$ is more difficult to recommend. As $\texttt{ONA}^*_\gamma$ relies on expensive parameter tuning of the gain factor γ, it is less time efficient and therefore less suitable in situations in which time is a limiting factor.

Overview

Table 7.2: Key properties of all MONDRIAN-based microaggregation heuristics presented in this chapter.

	Time	Space	Introduced	Reference
MONDRIAN	$O(nd\log n)$	$O(nd)$	2006	[37]
MONDRIAN_V	$O(nd\log n)$	$O(nd)$	2021	[71]
MONDRIAN_V2D	$O(nd^2\log n)$	$O(nd)$	2021	[71]
$\texttt{MONA}_\rho$	$O(n^{1+\rho}d)$	$O(nd)$	2021	[71]
$\texttt{MONA\_2D}_\rho$	$O(n^{1+\rho}d)$	$O(nd)$	2021	[71]

7.3 Experimental Evaluation of Near Linear Time Heuristics

For the experiments presented in this section, the same general test setting and benchmark databases as in 5.2 and 6.5 have been used. However, as algorithms aimed at large databases are evaluated, there is little use in benchmarking small databases of only a few thousand elements. Hence, MONDRIAN and MONA variants are tested on Credit Card and Adult only. Additionally, a new very large database called *Winnipeg* has been added to the test suite. Like Credit Card and Adult, the Winnipeg database is from the UCI machine learning repository [42] in which it is called *Crop mapping using fused optical-radar data set*. It

consists of 325834 elements in 174 numeric dimensions and has been created by combining satellite and radar images of cropland near the Canadian city of Winnipeg for the use as a benchmark database for classification algorithms. Because the Winnipeg database is too large to be handled by quadratic time heuristics like ONA*, it is especially useful to measure in which way MONDRIAN variants widen the applicability of microaggregation and to observe how far MONA can reduce information loss in comparison to MONDRIAN variants.

A comparison of information losses between the different variants of MONDRIAN, MONA and ONA* is given in table 7.3. It can be seen that MONDRIAN_V is able to lower information losses to about 85% of those achieved by MONDRIAN. As this improvement comes at no extra costs regarding time or space requirements we can consider MONDRIAN_V as a new baseline upon which other heuristics have to improve. Adding two-dimensional splits further decreases information loss, but comes at extra cost. In the case of Adult (see table 7.4) the additional time consumption as well as the information loss improvements of two-dimensional splits are small. This comes at no surprise as Adult consists of only three-dimensional data. Far greater improvements are possible for Credit Card (see table 7.5) which contains 24-dimensional data. However, this improvement comes at the cost of about 10 times higher time consumption, caused by the d^2 factor within the time complexity of MONDRIAN_V2D. On even higher dimensional data like Winnipeg (see table 7.6) the additional time consumption of two-dimensional splits becomes unjustified. Not only does the information loss decrease only slightly, the time consumption of MONDRIAN_V2D is even higher than that of $\text{MONA}_{0.7}$ which is far superior at preserving utility. All together MONDRIAN_V2D achieves about 89% of the information loss of MONDRIAN_V over all three databases and k from 2 to 10.

Regarding MONA, the aforementioned tables as well as figures 7.7,7.8 and 7.9 indicate large effects of even small split factors ρ. For $\rho = 0.5$ MONA causes only half as much information loss as MONDRIAN_V, while consuming 2 to 3 orders of magnitude less time compared to ONA*. If even faster heuristics are needed, e.g. for databases consisting of millions of elements, $\text{MONA}_{0.3}$ might be a valid option, as first significant improvements appear for $\rho \geq 0.3$. When there is more time, $\text{MONA}_{0.7}$ or $\text{MONA}_{0.8}$ are able to narrow the gap between the information loss of MONA and ONA* further, while still being significantly faster than ONA*.

Two aspects have to be considered when adding two-dimensional splitting to MONA. First of all, the influence of two-dimensional splitting decreases with larger ρ, as less splits are performed until ONA* takes over. Besides that, the same general behavior as for MONDRIAN_V2D can be observed. If MONDRIAN_V2D profits from two-dimensional splitting, so will MONA_2D.

An irregularity one might observe from the data is the consistency of information losses of MONDRIAN variants over different k on the same database. Whereas other heuristics suffer from increased information loss for larger k, this is not necessarily true for them. The reason is quite simple: As MONDRIAN uses its splitting rule until cluster sizes of less than $2k$ are achieved, there might not be a difference between e.g. $k = 2$ and $k = 3$ for different sizes n of databases. Take for example Credit Card, which emits this behavior for all three MONDRIAN variants and $k \in \{2, 3\}$. After 12 rounds of splitting, the average cluster size is about 7.3, allowing for another split for $k = 2$ as well as for $k = 3$. However, after 13 rounds of splitting, the average cluster size falls to 3.66 which prevents another split for both values of k, effectively creating the same clustering for both anonymity factors k.

Table 7.3: Average percentage information loss difference between a selection of heuristics over Adult, Credit Card and Winnipeg. A value below 100 indicates that an algorithm causes less information loss on average. Differences based on ONA* do not include data from the Winnipeg database, as it is too big to be handled with ONA* within a reasonable amount of time.

	Information Loss Differences in %					
from\to	MONDRIAN	MONDRIAN_V	MONDRIAN_V2D	$\text{MONA}_{0.5}$	$\text{MONA\_2D}_{0.5}$	ONA*
MONDRIAN	100	85.1	75.8	45.8	43.1	22.6
MONDRIAN_V		100	88.5	53.2	49.9	28.2
MONDRIAN_V2D			100	60.7	56.6	33.3
$\text{MONA}_{0.5}$				100	93.8	61.4
$\text{MONA\_2D}_{0.5}$					100	66.1
ONA*						100

Table 7.4: Information losses and runtimes of MONDRIAN variants, ONA* and a selection of MONA heuristics on Adult $\in \mathcal{X}_{48842,3}$.

	Information Loss in % on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	0.249	0.508	0.508	0.508	0.917	0.917
MONDRIAN_V	0.206	0.407	0.407	0.407	0.757	0.757
MONDRIAN_V2D	0.188	0.387	0.387	0.387	0.705	0.705
$\text{MONA}_{0.3}$	0.067	0.140	0.211	0.285	0.434	0.649
$\text{MONA\_2D}_{0.3}$	0.068	0.142	0.216	0.290	0.425	0.646
$\text{MONA}_{0.5}$	0.050	0.106	0.161	0.211	0.322	0.465
$\text{MONA\_2D}_{0.5}$	0.050	0.102	0.156	0.206	0.302	0.456
$\text{MONA}_{0.7}$	0.043	0.091	0.137	0.182	0.276	0.402
$\text{MONA\_2D}_{0.7}$	0.042	0.090	0.137	0.182	0.268	0.399
ONA*	0.039	0.081	0.123	0.165	0.243	0.357

	Runtime in s on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	0.11	0.05	0.04	0.04	0.03	0.03
MONDRIAN_V	0.09	0.05	0.05	0.05	0.04	0.04
MONDRIAN_V2D	0.20	0.09	0.07	0.07	0.07	0.07
$\text{MONA}_{0.3}$	0.33	0.20	0.17	0.38	0.41	0.70
$\text{MONA\_2D}_{0.3}$	0.36	0.22	0.20	0.40	0.37	0.69
$\text{MONA}_{0.5}$	0.65	0.42	0.49	0.61	0.76	0.77
$\text{MONA\_2D}_{0.5}$	0.61	0.45	0.52	0.69	0.79	0.72
$\text{MONA}_{0.7}$	4.45	2.86	2.85	2.76	2.47	2.10
$\text{MONA\_2D}_{0.7}$	4.54	3.03	3.14	2.90	2.62	2.16
ONA*	226.28	148.64	116.42	99.75	78.56	52.26

Table 7.5: Information losses and runtimes of MONDRIAN variants, ONA* and a selection of MONA heuristics on Credit Card $\in \mathcal{X}_{30000,24}$.

	Information Loss in % on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	30.33	30.33	41.47	41.47	43.75	50.71
MONDRIAN_V	24.05	24.05	32.54	32.54	34.12	39.27
MONDRIAN_V2D	17.85	17.85	24.52	24.52	25.78	30.30
$\text{MONA}_{0.3}$	10.85	17.35	22.26	25.63	31.67	39.27
$\text{MONA\_2D}_{0.3}$	8.76	13.92	17.87	20.50	24.84	30.30
$\text{MONA}_{0.5}$	7.74	12.56	15.99	18.53	22.45	26.59
$\text{MONA\_2D}_{0.5}$	6.87	10.96	13.89	16.16	19.50	22.95
$\text{MONA}_{0.7}$	5.64	9.14	11.52	13.41	16.27	19.22
$\text{MONA\_2D}_{0.7}$	5.27	8.47	10.75	12.42	14.97	17.54
ONA*	3.50	5.86	7.53	8.65	10.23	12.24

	Runtime in s on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	0.14	0.09	0.06	0.06	0.05	0.05
MONDRIAN_V	0.16	0.07	0.06	0.06	0.07	0.06
MONDRIAN_V2D	1.12	1.18	1.04	1.04	1.02	0.97
$\text{MONA}_{0.3}$	0.34	0.20	0.17	0.13	0.14	0.06
$\text{MONA\_2D}_{0.3}$	1.21	1.14	1.07	0.98	1.03	0.96
$\text{MONA}_{0.5}$	0.49	0.36	0.31	0.33	0.36	0.46
$\text{MONA\_2D}_{0.5}$	1.19	1.10	1.04	1.07	1.11	1.21
$\text{MONA}_{0.7}$	1.64	1.14	1.08	1.06	1.06	1.25
$\text{MONA\_2D}_{0.7}$	1.94	1.79	1.62	1.56	1.62	1.75
ONA*	49.27	42.92	35.37	36.49	32.84	29.62

Table 7.6: Information losses and runtimes of MONDRIAN variants and a selection of MONA heuristics on Winnipeg $\in \mathcal{X}_{325834,174}$.

	Information Loss in % on Winnipeg					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	49.43	65.33	65.33	75.71	75.71	81.56
MONDRIAN_V	46.53	61.93	61.93	72.51	72.51	78.72
MONDRIAN_V2D	44.26	59.60	59.60	70.24	70.24	76.70
$\text{MONA}_{0.3}$	30.01	44.15	52.50	57.98	64.91	70.93
$\text{MONA\_2D}_{0.3}$	29.37	42.96	50.98	56.32	63.09	69.03
$\text{MONA}_{0.5}$	22.25	33.47	41.04	46.84	55.09	62.56
$\text{MONA\_2D}_{0.5}$	21.76	32.58	39.87	45.52	53.72	61.07
$\text{MONA}_{0.7}$	15.64	23.38	28.30	31.68	36.34	40.74
$\text{MONA\_2D}_{0.7}$	15.34	22.85	27.38	30.59	34.94	39.01

	Runtime in s on Winnipeg					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	6.8	6.4	6.2	5.9	5.8	5.6
MONDRIAN_V	9.4	8.8	8.7	8.7	8.7	8.3
MONDRIAN_V2D	1407.7	1475.3	1483.6	1485.4	1416.4	1323.5
$\text{MONA}_{0.3}$	16.6	16.8	16.5	17.0	20.1	19.3
$\text{MONA\_2D}_{0.3}$	1489.9	1387.4	1342.1	1356.5	1394.2	1396.8
$\text{MONA}_{0.5}$	45.7	42.0	41.0	40.6	42.3	51.0
$\text{MONA\_2D}_{0.5}$	1533.6	1223.9	1239.4	1208.0	1298.1	1285.7
$\text{MONA}_{0.7}$	721.3	646.2	591.8	554.9	495.8	453.6
$\text{MONA\_2D}_{0.7}$	1810.3	1572.3	1597.0	1501.8	1386.3	1354.9

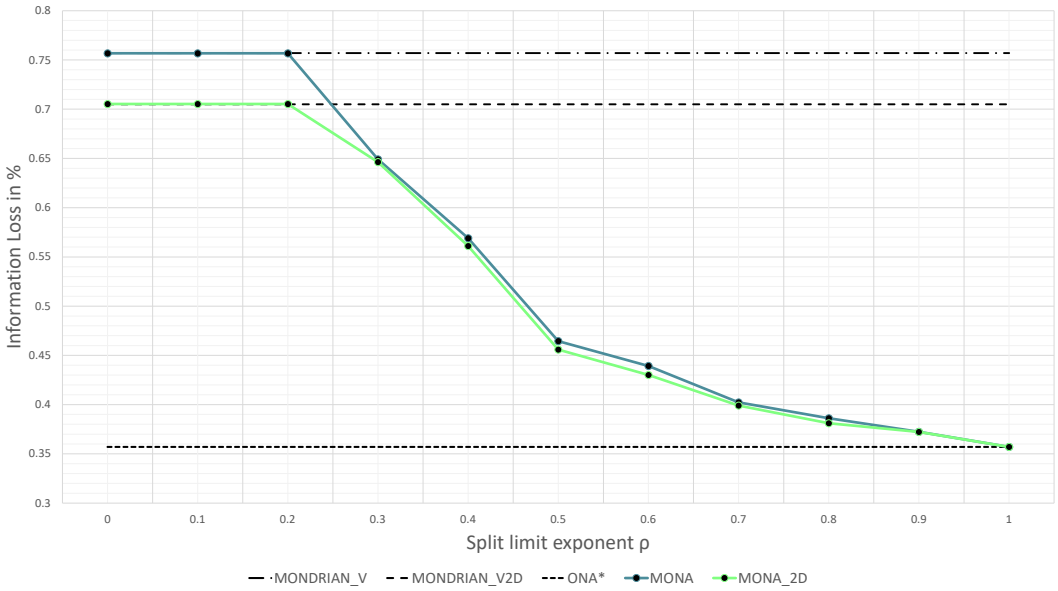


Figure 7.7: Information Losses of MONA and MONA_2D for different split limits compared to MONDRIAN_V variants and ONA* on the Adult database with $n = 48842$ and $d = 3$ for $k = 10$. As $n^\rho < 2k$ for $\rho \leq 0.2$, both MONA and MONA_2D behave like their MONDRIAN_V counterparts for these ρ.

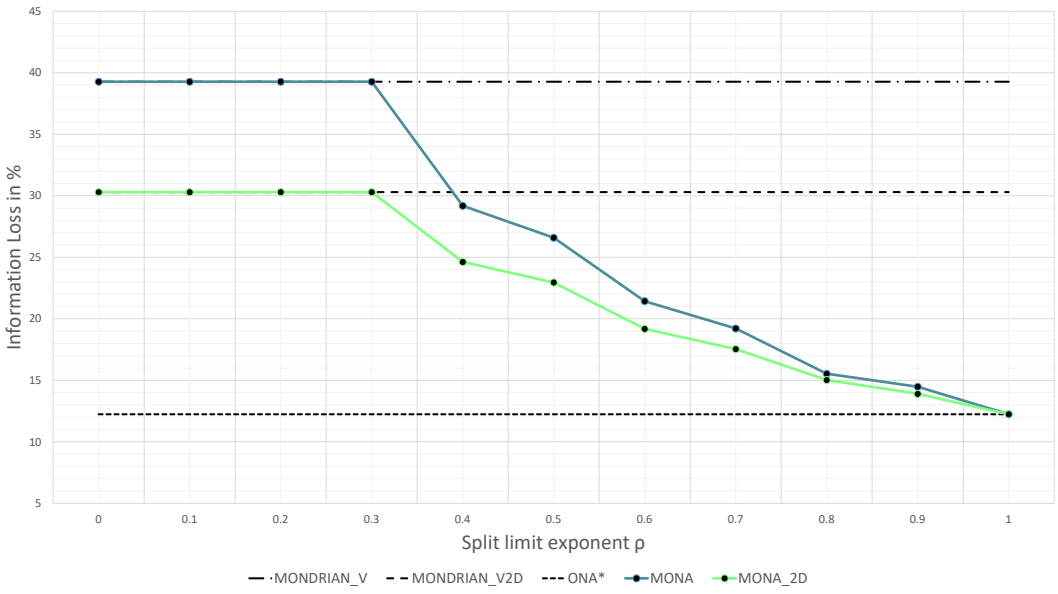


Figure 7.8: Information Losses of MONA and MONA_2D for different split limits compared to MONDRIAN variants and ONA* on the Credit Card database with $n = 30000$ and $d = 24$ for $k = 10$. As $n^\rho < 2k$ for $\rho \leq 0.3$, both MONA and MONA_2D behave like their MONDRIAN_V counterparts for these ρ.

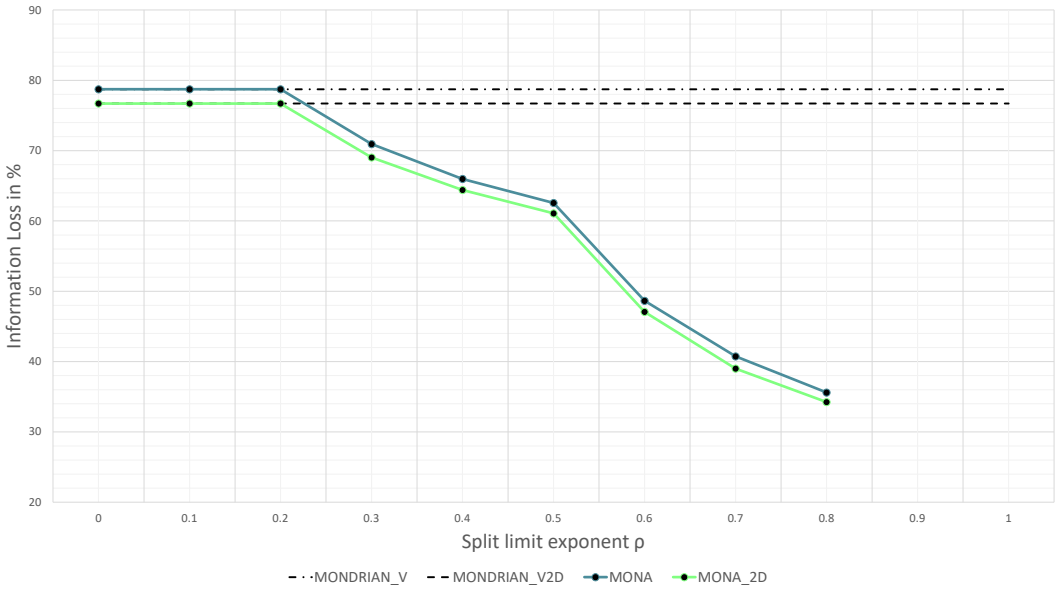


Figure 7.9: Information Losses of MONA and MONA_2D for different split limits compared to MONDRIAN variants on the Winnipeg database with $n = 325834$ and $d = 174$ for $k = 10$. As $n^\rho < 2k$ for $\rho \leq 0.3$, both MONA and MONA_2D behave like their MONDRIAN_V counterparts for these ρ.

8

Additional Techniques used in Microaggregation Heuristics

There are many microaggregation heuristics based on different principles that have not been covered so far, mainly because they do not offer competitive information losses. Nevertheless, a full analysis of microaggregation heuristics would not be complete without looking at ways to create k-member clusterings by other means than maximum distance, applying Lloyd's algorithm or Mondrian splitting. In this chapter we take a short look at other microaggregation heuristics with some unique properties that separate them from more established ones.

8.1 Minimum Spanning Tree Approach

In 2005 Laszlo and Mukherjee introduced a microaggregation heuristic based on minimum spanning trees [36]. Similar to approaches altering Lloyd's algorithm, they used a standard k-means clustering approach and adapted it to create a k-member clustering.

To create a k-means clustering by MST, elements are considered to be vertices of a complete weighted graph $G = (V, E, w)$ with Euclidean distances as edge weights. An MST is created on G and the *longest* (i.e. with highest weight) $k - 1$ edges are removed. The resulting forest contains exactly k trees, each of which represents a cluster. As long edges represent great distance between elements, the technique is suitable to create a clustering of low information loss.

The adapted process to create k-member clusterings is quite similar. After the creation of a global MST, edges are sorted in descending order according to their weight (Euclidean distance). Going through all edges, an edge is removed if and only if both resulting trees have at least k vertices. After all edges are considered, we obtain a forest that can easily be transformed into a k-member clustering. As the process does not guarantee an upper bound for the cluster size[6], clusters of size $\geq 2k$ are suggested to be subsequently split by applying maximum distance heuristics.

Time complexity of the algorithm is $O(n^2)$ which is needed both for creating an MST and to decide which edges can or cannot be cut without violating the minimum size constraint. No approximation guarantees are given and experimental data obtained by the

[6] Consider e.g. a star topology, which does not allow the removal of any edge.

authors suggests that information loss is similar but not superior to that of MDAV.

8.2 Two Fixed Reference Points

TFRP is an incremental microaggregation clustering approach similar to maximum-distance based heuristics. Introduced by Chang et al. in 2007 [13] TFRP calculates two artificial elements R_1 and R_2 by finding minimum and maximum attribute values $G_{\min}(X)$ and $G_{\max}(X)$ over all elements and dimensions, i.e. $R_1 = (G_{\min}(X), \ldots, G_{\min}(X)) \in \mathcal{R}^d$, $R_2 = (G_{\max}(X), \ldots, G_{\max}(X)) \in \mathcal{R}^d$. Cluster-generating elements x_r and x_s are found in greatest distance to G_1 and G_2. While G_1 and G_2 stay fixed during the run of the algorithm, in each clustering round new x_r and x_s are selected and clustered with their respective $(k-1)$ nearest neighbors. When less than k elements remain, unassigned elements are assigned to their closest clusters.

Unlike maximum-distance heuristics, TFRP has a second phase aimed at further reducing information loss. For this, clusters are sorted in descending order according to their cluster cost and a dissolve check is applied to each cluster in order. When the information loss can be lowered by reassigning all of its elements, a cluster is dissolved, just as in the dissolve phase of ONA and ONA*. After each cluster has been processed, clusters of $2k$ or more elements are split by applying the first phase of TFRP on these clusters.

While the performance is similar to MDAV$^+$(both have time complexity $O(n^2dk^{-1})$), TFRP fails at delivering consistently lower information loss than MDAV-variants. Nevertheless, TFRP can be seen as a first attempt at including dissolve and split mechanisms to microaggregation heuristics which eventually led to modern implementations like ONA*. The heuristic DBA presented in the next section can be seen as an alternative to TFRP lowering information loss at the cost of consuming significantly more time, while relying on the same post-processing approach. Experimental data on both, compared to MDAV* and ONA*, is included in table 8.1.

8.3 Density-based Microaggregation

Density-based Microaggregation (DBA) is an algorithm including techniques from MDAV as well as from TFRP, while being neither a maximum-distance nor a Lloyd-based heuristic. It has been introduced by Lin et al. in 2010 [44] and predates modern Lloyd-based microaggregation heuristics.

In the first phase of DBA, clusters are created sequentially like in maximum-distance heuristics. However, instead of using outlier elements as cluster-generating elements, clusters are built around elements with highest density within unassigned elements. This process is repeated until less than k elements remain unassigned. After that, remaining elements are assigned to their closest clusters. In its second phase DBA tries to lower information loss by dissolving clusters in inverse order of their original creation. Although applied in a much simpler way, this check is identical to the dissolve check used by ONA and ONA*. Finally, in

a third phase, DBA splits large clusters by applying the MDAV heuristic. See algorithm 21 for a complete description of DBA.

Although TFRP and DBA use similar split and dissolve mechanisms to ONA, the overall procedure is quite distinct from Lloyd-based heuristics. TFRP and DBA operate in a single run of creating an initial clustering and optimizing it afterwards. A cluster is either dissolved or it is not, there is no reconsideration after changes in other parts of the clustering. Further, the possibility of reassigning single elements is not used. Experimental data provided by the authors of DBA indicate that information loss is on par with that of MDAV* but fails to undercut it reliably. See table 8.1 for a comparison between TFRP, DBA, MDAV* and ONA*.

The authors did not give complexity results or experimental data on time consumption. To compute cost of k-neighborhoods for each element x, $O(n^2kd)$ time is needed. As after each cluster creation the set of unassigned elements U changes, neighborhoods and clusters have to be updated. Depending on the data structure used, this update process might be more or less efficient. However, as $O(n/k)$ clusters have to be created, there are $O(n/k)$ update cycles which are associated with some cost. Hence an optimistic approximation of time complexity for DBA is $O(n^3d)$, which is too high for large databases.

Table 8.1: Comparison of information losses of TFRP, DBA, MDAV* and ONA* based on data on TFRP provided in [13] and data on DBA provided in [44]. Only on EIA DBA is able to deliver better results than MDAV*.

	Information Loss in %			
	$k = 3$	$k = 4$	$k = 5$	$k = 10$
MDAV* on Census	5.78	7.45	8.83	14.00
TFRP on Census ([13])	5.80	7.64	8.98	13.96
DBA on Census ([44])	5.58	7.59	9.05	13.52
ONA* on Census	5.27	6.71	8.04	12.46
MDAV* on Tarragona	16.15	19.19	22.26	34.74
TFRP on Tarragona ([13])	16.88	19.18	21.85	33.01
DBA on Tarragona ([44])	16.15	22.67	25.45	34.81
ONA* on Tarragona	15.11	17.79	20.48	31.15
MDAV* on EIA	0.45	0.62	0.91	2.63
TFRP on EIA ([13])	0.43	0.60	0.91	2.59
DBA on EIA ([44])	0.42	0.56	0.82	2.08
ONA* on EIA	0.37	0.52	0.79	1.99

8.4 Sorting-based Microaggregation

Another interesting approach has been presented by Mahmood, Kabir and Mustafa in 2012 [49]. The core idea of this so-called *sorting-based microaggregation* (SBM) is to sort a database X by a specific multi-dimensional sorting algorithm and afterwards repeatedly create clusters around the smallest and largest elements. Similar to MD, SBM tries to find elements in greatest distance to build its clusters. Before discussing some crucial drawbacks and oversights of SBM let us take a detailed look at its core functionality.

Algorithm 21: DBA [44]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

Let $U \leftarrow X$
Let $\mathcal{C} \leftarrow \emptyset$
Let counter $\leftarrow 0$
repeat
  foreach $x \in U$ **do**
    Let $cost(x) \leftarrow cost(N_{k-1}(x, U))$
  Let $x_r \leftarrow \arg\min_{x \in U} cost(x)$
  Let $C_{\text{counter}} \leftarrow N_{k-1}(x_r, U)$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_{\text{counter}}\}$
  $U \leftarrow U \setminus C_{\text{counter}}$
  counter $\leftarrow$ counter $+ 1$
until $|U| < k$
foreach $x \in U$ **do**
  $clos(x) \leftarrow clos(x) \cup \{x\}$
for $i \leftarrow (counter - 1)$ *down to* 0 **do**
  // Should cluster C_i be dissolved?
  foreach $s \in C_i$ **do**
    Let $C_{\text{target}}(s) \leftarrow \arg\min_{C \in \mathcal{C} \setminus \{C_i\}} \delta(s, c(C))^2$
  Let $\mathcal{C}_{\text{target}} \leftarrow \bigcup_{s \in C_i} \{C_{\text{target}}(s)\}$
  Let $\mathcal{C}'_{\text{target}} \leftarrow \emptyset$
  foreach $C \in \mathcal{C}_{\text{target}}$ **do**
    $\mathcal{C}'_{\text{target}} \leftarrow \mathcal{C}'_{\text{target}} \cup \{C \cup \{s \in C_i \mid C_{\text{target}}(s) = C\}\}$
  Let $cost_K \leftarrow cost(C_i) + cost(\mathcal{C}_{\text{target}})$
  Let $cost_D \leftarrow cost(\mathcal{C}'_{\text{target}})$
  if $cost_K > cost_D$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \setminus (\{C_i\} \cup \mathcal{C}_{\text{target}})$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'_{\text{target}}$
foreach $C \in \mathcal{C}$ **do**
  if $|C| \geq 2k$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \texttt{MDAV}(C)$

While the sorting mechanism in the original paper is described in a more complicated way, its effect can be reduced to the following process: For each dimension j consider a sorted list of all x_i in ascending order of their values. To each element x_i assign the sum of its ranks within these sorted lists. The resulting sorted database X' is a permutation of elements in ascending order of their sum of ranks from the previous step. Ambiguities are broken up arbitrarily. See table 8.2 for an example of this process.

Table 8.2: Example of the sorting mechanism used by SBM. Table (a) is the input database X, table (b) describes intermediate steps and table (c) is the resulting sorted database X'.

Element	Attribute 1	Attribute 2
x_1	7	6
x_2	6	3
x_3	9	7
x_4	8	1

(a) Input database X

Element	Rank 1	Rank 2	Rank Sum	New Index
x_1	2	3	5	3
x_2	1	2	3	1
x_3	4	4	8	4
x_4	3	1	4	2

(b) Intermediate steps

Element	Attribute 1	Attribute 2
x'_1	6	3
x'_2	8	1
x'_3	7	6
x'_4	9	7

(c) Sorted database X'

After the database is sorted, two clusters are created. First, the first element x'_1 is clustered with its $(k-1)$-nearest neighbors and afterwards the last element x'_n is clustered with its $(k-1)$-nearest neighbors. Subsequently the database of remaining elements is sorted and reduced again and again until less than $3k$ elements remain. If at least $2k$ elements are left, the database is sorted one last time but only one cluster is created around the smallest element. Remaining elements are clustered together in a single cluster of up to $2k-1$ elements.

Despite the novel idea of sorting-based microaggreagtion, SBM fails to compete with maximum-distance heuristics like MDAV$^+$. Furthermore, the original publication includes false assumptions, missing information and questionable experimental results:

– The authors claim that after sorting, the smallest and largest elements are in greatest (Euclidean) distance to each other among all pairs of elements. The database depicted in table 8.3 is a counterexample to this claim. While the smallest element is x_1 and the largest is either x_2 or x_4, the greatest distance is between x_3 and x_4.

Table 8.3: Exemplary database disproving the claim of maximum (Euclidean) distance between the smallest and largest elements of sorted databases according to SBM.

Element	Attribute 1	Attribute 2
x_1	-1	-1.5
x_2	1	1
x_3	-2	1.5
x_4	2	-1

- The authors propose to sort after each round of creating two clusters. However, as the order of elements does not change when elements are removed, there is no need to sort more than once.
- There is no time complexity analysis stated by the authors. It seems to be $O(nd \log n)$ for sorting and $O(n^2/k\ d)$ for nearest-neighbor selection.
- The authors claim significantly lower information losses on standard databases compared to MDAV and other heuristics. While there is no public reference implementation, a re-implementation within the test suite of this thesis indicates significantly higher information losses than stated by the authors (see table 8.4).
- The authors did not indicate what hardware and programming language was used to obtain runtime results and their execution times reach minutes on EIA, which is extraordinary high even accounting for unnecessary sorting and typical hardware from 2012 when compared to similar maximum-distance algorithms discussed above. A comparison between original data and data from the test setting used throughout this dissertation is given in table 8.4.

Table 8.4: Comparison of information loss and runtime results of SBM claimed in [49] and based on re-implementation.

	Information Loss in %			
	$k = 3$	$k = 4$	$k = 5$	$k = 10$
MDAV$^+$ on Census	5.66	7.51	9.01	14.07
SBM on Census ([49])	2.10	3.63	3.46	6.85
SBM on Census	6.47	9.47	10.58	16.66
MDAV$^+$ on Tarragona	16.95	19.77	22.87	33.25
SBM on Tarragona ([49])	9.86	12.00	18.17	32.13
SBM on Tarragona	18.71	22.98	28.25	32.59
MDAV$^+$ on EIA	0.49	0.67	1.78	3.55
SBM on EIA ([49])	0.40	0.53	0.80	1.77
SBM on EIA	0.63	0.87	2.19	4.41

	Runtime in s			
	$k = 3$	$k = 4$	$k = 5$	$k = 10$
MDAV$^+$ on EIA	0.06	0.05	0.04	0.02
SBM on EIA ([49])	230	250	270	310
SBM on EIA	0.10	0.09	0.08	0.07

8.5 Summary

Since 2005, several new heuristics have been introduced with the goal of replacing maximum distance based microaggregation heuristics. So far, only the Lloyd-based approach presented in chapter 6 has been successful in achieving this objective. Nevertheless, the approaches presented in this chapter might inspire new ideas for improvements able to deliver even better ratios between time consumption and information loss in the future.

9

Microaggregation Algorithms with Approximation Guarantees

As can be seen in the previous four chapters, a lot of work has been done, trying to create efficient microaggregation heuristics that work well in practice. However, for none of the heuristics discussed previously, approximation bounds are known. For the problem of k-means clustering there is a wide variety of polynomial time approximation algorithms, so one might ask why so little is known about approximations of the similar problem of k-anonymous microaggregation. In fact, only a single approximation algorithm with a rather high approximation factor of $O(k^3)$ is known so far [19]. While there is no obvious reason, why creating approximations for the microaggregation problem could be harder than for the k-means clustering problem, we can observe several problems that prevent a simple adaptation of k-means approximation schemes.

In this chapter we first take a look at μ-Approx, the only polynomial time microaggregation approximation known so far, include a short recap of k-means clustering approximations and finally explore opportunities and problems on a way to create better approximation algorithms for the microaggregation problem. But before we start, let us first recap the basic definitions of approximation algorithms.

Definition 9.1 (Optimization Problem).
An *optimization problem* $\Pi = (I, S, \sigma, f)$ is a tuple consisting of a set of *problem instances* I, a set of *solutions* S, a mapping $\sigma : I \to 2^S$ from problem instances $i \in I$ to sets of feasible solutions $\sigma(i) \subseteq S$ and an *objective function* $f : I \times S \to \mathbb{R}_0^+$, assigning a non-negative real value to pairs of problem instances and feasible solutions.

An *optimal solution* $OPT(i)$ to an instance $i \in I$ is either a feasible solution of minimal or maximal objective function value, depending on whether Π is a *minimization* or *maximization* problem.

Definition 9.2 (α-Approximation Algorithm).
Given an optimization problem $\Pi = (I, S, \sigma, f)$, an algorithm $\mathcal{A}$ is an *α-approximation algorithm* for Π if for any instance $i \in I$ a solution $s \in \sigma(i)$ is computed in polynomial time[7], such that

- $f(s) \leq \alpha \cdot OPT(i)$, if Π is a minimization problem and
- $f(s) \geq OPT(i)/\alpha$, if Π is a maximization problem.

[7] More precisely: in time polynomial in the size of i.

Fact 9.3.
For each $n, d, k \in \mathbb{N}$, the optimal k-anonymous microaggregation problem is an optimization problem Π_μ with databases $X \in \mathcal{X}_{n\times d}$ as inputs, k-member clusterings $\mathcal{C}_X$ as feasible solutions and the distortion cost function *cost* as objective function. The optimal solution to a database X is the k-member clustering $\mathcal{C}_X$ of minimal cost.

9.1 Existing Microaggregation Approximation

The currently only existing microaggregation approximation algorithm called μ-Approx has been introduced by Domingo-Ferrer et al. in 2008 [19]. In a way it is similar to the minimum spanning tree approach presented in section 8.1, as in both algorithms elements are considered as vertices of a weighted graph with edge weights describing distances between elements. However, in μ-Approx a weighted directed acyclic graph is used instead of a forest of trees.

μ-Approx takes a database $X \in \mathcal{X}_{n\times d}$ and an anonymity parameter k and creates a weighted directed acyclic graph (DAG) $G(X,k) := (V,E,w)$ for $V := \{1,\dots,n\}$, $E \subseteq \{(u,v)|u,v \in V\}$ and $w(u,v) := \delta(x_u, x_v)$. The set of incoming edges of a node $u \in V$ is noted $\texttt{in}(u)$, analogously the set of outgoing edges of a node u is noted $\texttt{out}(u)$. *Connected components* $C \subseteq V$ of $G(X,k)$ are non-extendable sets of nodes that are connected by edges. As nodes of V are indices of elements within X, the set $\mathcal{C}$ of connected components can also be interpreted as a clustering according to definition 2.9. As defined in chapter 5 we use $N_\ell(u, V')$ to denote the set of $u \in V$ and the indices of the ℓ nearest neighbors of x_u according to $\delta^2(\cdot,\cdot)$ restricted to the set of elements x_v with $v \in V' \subseteq V$. A solution to the k-anonymous microaggregation problem can be obtained by considering each connected component C as a cluster of its elements.

The algorithm μ-Approx consists of three sub-procedures called in order and only once. After that, the following properties regarding the resulting graph are guaranteed:

1. Each vertex has at most one outgoing edge.
2. An edge (u,v) exists only if v is one of the $k-1$ nearest neighbors of u according to δ^2.
3. The size of every connected component is at least k.
4. The size of every connected component is at most $\max(2k-1, 3k-5)$.

The first sub-procedure of μ-Approx called Forest creates a DAG that satisfies condition 1 to 3 but not the upper limit of component size stated in condition 4. Next, in the sub-procedure called Decompose, components of size $s > \max(2k-1, 3k-5)$ are split up to satisfy condition 4. Finally, MDAV is used on all components that have $2k$ or more elements. As there are between $2k$ and $3k-5$ elements, MDAV creates exactly 2 clusters, of which one contains k and the other one all remaining elements. Forest and μ-Approx are shown in algorithm 22 and algorithm 23, respectively. A pseudocode of MDAV is given in algorithm 2 in chapter 5. Decompose consists of a lengthy case analysis which describes, how all connected components of size $s > \max(2k-1, 3k-5)$ are split in total time $O(n^2)$ without violating condition 3 in the resulting DAG. As its details are rather technical and

not needed to understand the approximation guarantees of μ-Approx, we will omit them here. Consider the original publication for details.

Algorithm 22: Forest [19]

input : database X of size n and minimal cluster size k
output: DAG $G = (V, E, w)$ with connected components $\mathcal{C} = \{C_1, \ldots, C_m\}$

1 Let $G = (V, E, w)$ with $V \leftarrow \{1, \ldots, n\}$ and $E \leftarrow \emptyset$
2 Let $\mathcal{C}$ be the set of connected components in G
3 **repeat**
4 Select any $C \in \mathcal{C}$ with $|C| \leq k$
5 Select any $u \in C$ with $\texttt{out}(u) = 0$
6 Select any $v \in N_{k-1}(u, V \setminus C)$
7 $E \leftarrow E \cup (u, v)$
8 Let $w(u, v) \leftarrow \delta(x_u, x_v)$
9 Update $\mathcal{C}$
10 **until** $\min_{C \in \mathcal{C}} |C| \geq k$

Algorithm 23: μ-Approx [19]

input : database X and minimal cluster size k
output: k-member clustering $\mathcal{C} = \{C_1, \ldots, C_m\}$

1 $G \leftarrow \texttt{Forest}(X, k)$
2 $G \leftarrow \texttt{Decompose}(G, k)$
3 Let $\mathcal{C}$ be the set of connected components in G
4 **foreach** $C \in \mathcal{C}$ **do**
5 **if** $|C| \geq 2k$ **then**
6 $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C\}$
7 $\mathcal{C} \leftarrow \mathcal{C} \cup \texttt{MDAV}(C, k)$

Theorem 9.4 ([19]).
Given $k \leq \sqrt{n}$, the worst case time complexity of algorithm 23 is $O(n^2)$.

Theorem 9.5 ([19]).
μ-Approx *is a $2(2k-1)(\max(2k-1, 3k-5))^2$-approximation algorithm for the k-anonymous microaggregation problem.*

Proof sketch. By theorem 2.19 we know that step 4 to 7 of algorithm 23 cannot increase cost. The approximation guarantee is shown to hold after step 2 already. Let $E(C)$ describe all edges in G between nodes from connected component $C \in \mathcal{C}$, further let $cost(\mathcal{C})$ indicate the sum of costs of the clusters corresponding to the components $\mathcal{C}$ and $\mathcal{C}_{OPT}$ a k-member clustering of optimal cost. Both $\mathcal{C}$ and $\mathcal{C}_{OPT}$ refer to the same database clustered. Two

inequalities are shown:

$$cost(\mathcal{C}) \leq \max(2k-1, 3k-5) \sum_{C \in \mathcal{C}} \left(\sum_{e \in E(C)} w(e) \right)^2 \tag{9.1}$$

$$\sum_{C \in \mathcal{C}} \left(\sum_{e \in E(C)} w(e) \right)^2 \leq 2(2k-1) \max(2k-1, 3k-5) cost(\mathcal{C}_{OPT}) \tag{9.2}$$

The combination of equation 9.1 and 9.2 shows the desired approximation bound. □

While an approximation guarantee of about $O(k^3)$ is quite high, in most cases μ-Approx seems to create clusterings with utility comparable to that of clusterings created by maximum distance heuristics. Nevertheless, as empirical results in [19] show, for practical applications, MDAV seems to be the better choice.

9.2 Overview of k-Means Approximations

In section 4.1 similarities and differences between the k-means and k-member clustering problems are discussed. We have seen that despite the similarities, k-means seems to be somewhat easier, both in terms of theoretical complexity classes as well as in properties limiting the search space for optimal solutions. In short, we have seen that optimal k-means clusterings are always multiset-respecting (see corollary 4.5) and that in any optimal k-means clustering each element is in the cluster represented by the closest centroid to it (see theorem 4.3). Both properties are missing in the k-member clustering problem. Unlike k-anonymous microaggregation, k-means is further known to be solvable in polynomial time if both d and k are fixed.

The wide availability of approximation algorithms for the k-means problem further seems to indicate that k-means is easier to approximate than microaggregation for which there is only a single non-trivial approximation algorithm, as described above. While it is out of scope for this thesis to discuss k-means approximation algorithms in detail, we now take a short look at some important results to see what can be learned from k-means approximations in respect of finding approximations for the microaggregation problem.

Coresets

A common approach used throughout many k-means approximation algorithms is the reduction of search space for cluster centers to finite sets of candidates. Introduced by Matušek in 2000 [52] and improved by Har-Peled and Mazumdar in 2004 [32] as well as by Feldman et al. in 2007 [27] this technique is mainly used to create $(1+\epsilon)$-approximation algorithms for k-means instances with constant k. While the details on how to create such coresets differ, the main idea is to guarantee that the cost of an optimal clustering restricted to choose

cluster representatives from the coreset, is not higher than $(1+\epsilon)$ times the cost of an unrestricted optimal clustering. This is guaranteed by placing center candidates on intersections of integer grid lines or placing them around centroids of another k-means approximation solution. Guarantees are shown by relying on the fact that for k-means clustering, each element is contained in the cluster of its closest center.

Local Search

A well-known k-means approximation algorithm that achieves an approximation factor of $(9+\epsilon)$ in the general case in polynomial time has been introduced by Kanungo et al. in 2004 [35]. The algorithm mostly referred to as the *local search algorithm* uses a discretization of the d-dimensional space into a set S of candidates for cluster centers as presented in [52]. Next, an arbitrary set of k initial centers is chosen from S. After this initialization step, an iterative approach tries to lower distortion by replacing up to p current centers by other center candidates from S. When no such improvement is possible, the process reaches convergence and the algorithm holds. The algorithm is *local* in the sense that only a fraction of centers is swapped in each step, hence both clusterings before and after the swap are quite similar. The authors show that after convergence, the resulting clustering $\mathcal{C}$ achieves $cost(\mathcal{C}) \leq (3+2/p)^2\, cost(\mathcal{C}_{OPT})$ as long as $\mathcal{C}_{OPT}$ only uses centers from S. The stated approximation factor of $(9+\epsilon)$ holds for large values of p and includes inaccuracies caused by the discretization, approximate convergence to reach polynomial runtime and the difference between $(3+2/p)^2$ and 9. Again, as the proof heavily relies on the fact that any element is clustered with its closest center, it is not applicable for k-member clusterings.

k-Means++

As discussed in theorem 6.2, the k-MEANS++ algorithm offers expected cost bounded by $O(\log k)$ times the cost of an optimal algorithm. In contrast to other approaches in this section, this is achieved without relying on coresets but by cleverly picking initial cluster centers for Lloyd's algorithm. As we have already seen in section 6.1, this guarantee again relies on the fact that any element is clustered with its closest center, so it is not applicable for k-member clusterings.

LP Relaxation

A modern breakthrough in k-means approximation has been an LP-relaxation by Ahmadian et al. in 2019 [1]. There have been LP-relaxations to clustering algorithms before, however, as they typically rely on the triangle inequality, which does not hold for the squared Euclidean distance, they could not be adapted to k-means (see section 2.4).

Starting with the LP-relaxation of the discrete[8] k-means clustering problem depicted in figure 9.1, the algorithm of Ahmadian et al. achieves an approximation guarantee of $(6.357+\epsilon)$ for any $\epsilon \geq 0$. The algorithm is highly complex but in essence relies on a Lagrangian relaxation of the LP depicted in figure 9.1. For this, condition (4) is moved into

[8] For discretization the coreset technique from [27] is used.

$$
\begin{array}{lllrcll}
\min & \displaystyle\sum_{i\in S, j\in X} x_{ij}\cdot\delta(j,i)^2 & & & & & \\
\text{s.t.} & (1) & \sum_{i\in S} & x_{ij} & \geq & 1 & \forall j\in X \\
& (2) & & x_{ij} & \leq & y_i & \forall j\in X, i\in S \\
& (3) & \sum_{i\in S} & y_i & \leq & k & \\
& (4) & & x_{ij}, y_i & \geq & 0 & \forall j\in X, i\in S
\end{array}
$$

Figure 9.1: LP-relaxation of the discrete k-means clustering problem. There is a variable y_i for each center candidate $i \in S$, which indicates whether i is active or not. Further there are variables x_{ij} that indicate whether a database element $j \in X$ is included in the cluster belonging to the center candidate $i \in S$. Distances $\delta(\cdot,\cdot)^2$ do not use variables of the LP but are instead computed and stored beforehand.

the objective, changing the objective function into

$$\min \sum_{i\in S, j\in X} x_{ij}\cdot\delta(j,i)^2 + \lambda\left(\sum_{i\in S} y_i - k\right)$$

which helps transforming fractional LP-solutions into valid solutions using only integers for x and y. The resulting LP allows the activation of more or less than k centers, which needs to be prevented by adjusting the activation cost λ accordingly. Doing this in polynomial time is one of the major problems solved by Ahmadian et al.

9.3 New Approaches for Microaggregation Approximation

Adapting the method of Ahmadian et al. seems to be a valid approach to obtain better approximation guarantees than the $O(k^3)$ bound of μ-`Approx`. This section is to be understood as a first part of such an adaptation. First, a simple method for discretization is introduced as a replacement for more elaborate solutions in the original approach, which are not applicable to k-member clusterings. In a second step, we take a look at the adapted LP, which describes microaggregation instances. Future work should focus on the task of evaluating a Lagrangian relaxation of this LP and to build a redistribution algorithm that converts fractional LP solutions into valid clusterings.

Discretization of a k-Member Clustering Instance

From 2.14 we know that optimal centers of any clusters lie at the centroid of their elements. As we further know that any optimal cluster has between k and $2k-1$ elements, a trivial discretization of center candidates would be the set of centroids of all possible sets of k to $2k-1$ elements. Although this solution would guarantee that an optimal clustering can still be found, it is impractical because of the computational overhead of $\sum_{i=k}^{2k-1}\binom{n}{i}$ centroid computations and the handling of this large candidate set.

A simplified version of this strategy would only consider centroids of clusters of size 2, regardless of k, reducing the complexity to $(n^2 - n)/2$. This would guarantee that there are centroid candidates near and between any accumulation of elements. However, to obtain approximation guarantees we need every centroid chosen in an optimal solution not to be farther away than a fixed value ϵ from a centroid candidate, even in a worst case situation. As simple examples show, there are situations in which optimal centroids are arbitrarily far away from centroid candidates. Take for example the one-dimensional database of three elements $x_1 = 0, x_2 = x_3 = \alpha$ for any $\alpha \in \mathbb{N}$ and consider $k = 3$. The set of center candidates would be $S = \{\frac{\alpha}{2}, \alpha\}$ but the correct optimal centroid for the only cluster possible would be $c = \frac{2}{3}\alpha$ which is in distance $\frac{1}{6}\alpha$ from its closest center candidate. Hence, by scaling α we obtain an unbounded rise in additional cost ϵ when choosing centroids from S instead of taking optimal ones. Similar counterexamples exist for sets of centroids of other fixed numbers of elements, including the set of elements as center candidates.

A third and more realistic approach would be to construct an ϵ-grid between the extreme values of elements in each dimension and allow centroids at grid intersections. As a consequence, no potential optimal centroid can be farther than

$$\alpha := \sqrt{d \cdot \left(\frac{\epsilon}{2}\right)^2} = \frac{\sqrt{d}\epsilon}{2}$$

from any center candidate. By applying theorem 2.15 we obtain that additional cost is bounded by

$$\sum_{C \in \mathcal{C}} |C| \cdot \delta(c(C), \hat{c}(C))^2 \leq n \cdot \alpha^2 \in O\left(nd\epsilon^2\right)$$

compared to the cost of an optimal clustering with optimal centroid selection. Further, assuming $\Delta_j := (\max_{x \in X} x^j - \min_{x \in X} x^j)$ for each dimension j, the size of such a center candidate set would be

$$\prod_{1 \leq j \leq d} \left(\frac{\Delta_j}{\epsilon} + 1\right).$$

It should be obvious, that this simple approach is far from optimal and can easily be improved by adapting the density of grids in more populated areas. Nevertheless, it shows that there are indeed feasible solutions to reduce the set of centroids to a relatively small set of candidates in polynomial time and introducing only an arbitrarily small approximation error.

Adaptation of k-Means LP-Relaxation

The most important step in designing an LP relaxation of the k-anonymous microaggregation problem is of course the formulation of a suitable LP. A simple adaptation of the k-means LP of figure 9.1 results in an LP relaxation of the microaggregation problem. Note, that all integer solutions to the LP depicted in figure 9.2 are valid k-member clusterings and that an optimal integer solution is also an optimal solution to the microaggregation problem.

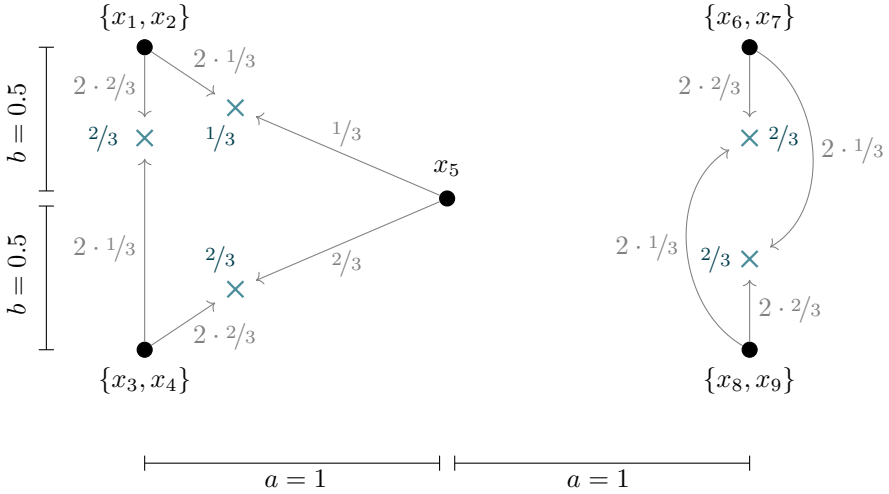
$$
\begin{array}{lllllll}
\min & \displaystyle\sum_{i\in S, j\in X} x_{ij}\cdot\delta(j,i)^2 & & & & & \\
\text{s.t.} & (1) & \sum_{i\in S} & x_{ij} & = & 1 & \forall j\in X \\
& (2) & & x_{ij} & \leq & y_i & \forall j\in X, i\in S \\
& (3) & \sum_{j\in X} & x_{ij} & \geq & k\cdot y_i & \forall i\in S \\
& (4) & & x_{ij}, y_i & \geq & 0 & \forall j\in X, i\in S \\
& (5) & & x_{ij}, y_i & \leq & 1 & \forall j\in X, i\in S
\end{array}
$$

Figure 9.2: LP-relaxation of the discrete k-anonymous microaggregation problem. There is a variable y_i for each center candidate $i \in S$, which indicates whether i is active or not. Further there are variables x_{ij} that indicate whether a database element $j \in X$ is included in the cluster belonging to the center candidate $i \in S$. Distances $\delta(\cdot,\cdot)^2$ do not use variables of the LP but are instead computed and stored beforehand.

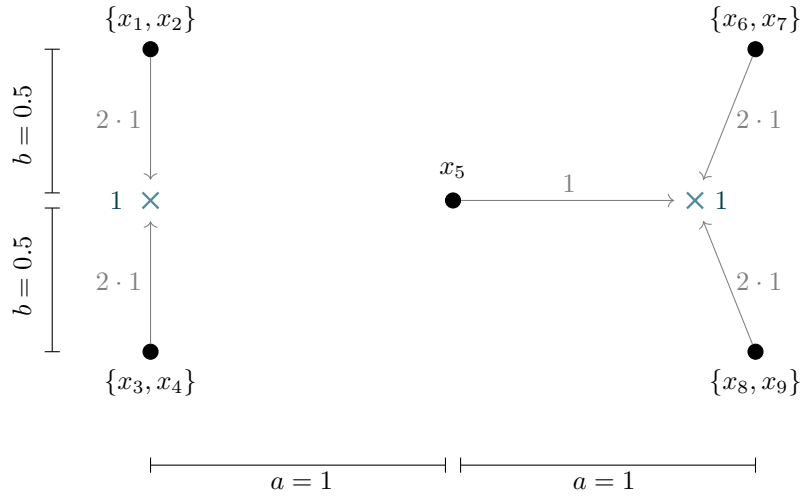
A key difference between microaggregation and k-member clusterings is the fact that cluster sizes for valid and optimal microaggregation solutions can be bounded by $k \leq |C| \leq 2k-1$. Hence, the solution set of the LP can safely be reduced by adding an additional condition

$$(6) \quad \sum_{j\in X} x_{ij} \leq (2k-1)y_i \quad \forall i \in S.$$

As it has been the case for solutions of the k-means clustering LP, it is mostly impossible to obtain a valid integer solution by rounding a fractional one. Not only can elements be *split up* between several clusters to circumvent cluster sizes of more than k, also clusters might be only partly active to allow less *element mass* being designated to them. Both problems can only be avoided by global redistribution algorithms, because re-allocations are needed and might propagate changes throughout the solution. A two-dimensional example comparing an optimal microaggregation solution with an optimal solution to the LP relaxation is shown in figure 9.3. Even though the *integrality gap*, i.e. the ratio between the values of the objectives of optimal integer and optimal fractional solutions is quite small in this example, there seems to be no easy way to exploit this. Instead, it can be seen that most element allocations and center activeness need to be changed to obtain any valid solution. As discussed above, a Lagrangian relaxation might help to reduce the work needed to obtain a valid clustering from a fractional LP solution.



(a) Optimal fractional LP solution



(b) Optimal integer LP solution

Figure 9.3: Example comparing an optimal (fractional) LP solution to an optimal integer valued solution representing an optimal 3-anonymous microaggregation for the given database $X = (x_1, \ldots, x_9)$ assuming a candidate set for centers generated by a 0.1-grid. Crosses indicate active centers and arrows (partial) membership of elements in clusters. Fractions at crosses and arrows indicate partial activeness of centers and partial membership of elements in clusters, respectively. The value of the objective function (equal to distortion cost for integer valued solutions) is 2.18 for solution (a) and 2.8 for solution (b).

10

Adapting Differential Privacy for Microdata Release

In 2006 Dwork el al. introduced differential privacy as a new – semantic privacy guarantee enabling private database queries [23]. Their work, published under the name *Calibrating Noise to Sensitivity in Private Data Analysis*, can be seen as one of the most influential inventions in modern privacy research, a feat for which the authors have been awarded the Gödel prize in 2017 [25].

Unlike syntactic privacy guarantees like k-anonymity, differential privacy limits disclosure of sensitive information regardless of attacker knowledge or data homogeneity, offering superior privacy compared to k-anonymity. It has been widely adopted in many research domains and in practice and is often seen as a new generation of privacy guarantee able to replace most previous ones. Until today, differential privacy is one of the most active research fields within computer and communication security.[9]

Recall, the topic of this thesis is the release of databases containing sensitive data from individuals complying with two conflicting requirements: On the one hand we want to protect the privacy of each individual contained in the database. On the other hand we want the published data to be useful for a broad range of scientific research without limiting its use case to specific analysis goals. While differential privacy could be used not only to anonymize database queries but also for the unrestricted release of data, there are many obstacles still to be solved for differential privacy to be able to replace microaggregation.

In this chapter we take a look at possible approaches to combine or replace microaggregation with or by differential private mechanisms. In the first section differential privacy and the two most important means to achieve it are defined concisely. After that we take a closer look at the achievements and shortcomings of existing approaches to combine microaggregation or k-anonymity with differential privacy. Finally, in the third section a path for further research is offered.

[9] As of March 2021 Google Scholar lists 152.000 results for *differential privacy* published in the year 2020 compared to 2000 for *k-anonymity*, 4000 for *post quantum cryptography*, 75.000 for *5G* and 273.000 for *corona virus*.

10.1 Differential Privacy in a Nutshell

In simple terms, differential privacy is a promise made by a data collector to its data providers (also called individuals). Individuals are guaranteed that any data published by the data collector is nearly identical, whether their data is included or not. As this promise is made to every data provider, it offers a protection not only in the average case but also in a worst case, i.e. independently of the specific data provided by the individuals. In other words, when data is released, each potential data provider can be assured that the (negative or positive) effects they have to face are nearly independent of their participation, i.e. whether they participate or not, the outcome is the same. Further, differential privacy offers *plausible deniability* to the data providers, as the influence of one individual is undetectable.

A common misconception about differential privacy is that it prevents any harm to data providers. However, such goal is unachievable as the objective of any study is a gain of information usable to affect the real world. When participants need to be shielded from any effect, no data, not even incorrect or fictional data may be published. As an example consider a study on census data which finds that dog owners are twice as likely to be involved in severe traffic accidents than a comparable group of cat owners. Whether this correlation is accurate or not, whether the results are found using differential private data or not and whether any particular dog owner's data is included in the data or not is irrelevant for the consequences any dog owner has to face when politics and insurance companies act on this result.

To define differential privacy more formally, we need some mathematical prerequisites. Definitions and results presented in this section follow the structure and notation used in the book *The Algorithmic Foundations of Differential Privacy* by Cynthia Dwork and Aaron Roth [24] as the techniques used in differential privacy are too different from microaggregation to adapt the notation presented in chapter 2.

Definition 10.1 (Probability Simplex).
Given a discrete set B, the *probability simplex* $\Delta(B)$ is

$$\Delta(B) = \left\{ P \in \mathbb{R}^{|B|} : p_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{|B|} p_i = 1 \right\}.$$

Definition 10.2 (Randomized Algorithm).
A *randomized algorithm* (*mechanism*) $\mathcal{M}$ with domain A, discrete range B and an internal mapping $M : A \to \Delta(B)$ returns a value $b = \mathcal{M}(a)$ with probability $(M(a))_b$ for each $b \in B$.

In differential privacy, databases are represented as histograms $X \in \mathbb{N}^{|\mathcal{X}|}$ in which each entry x_i represents the number of elements of type $i \in \mathcal{X}$ within X. The size of a database can be computed as $||X||_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i|$.

Definition 10.3 (Distance Between Databases).
The (ℓ_1-)*distance between two databases* X and Y is

$$||X - Y||_1 := \sum_{i=1}^{|\mathcal{X}|} |x_i - y_i|.$$

Definition 10.4 (Differential Privacy [24]).
A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ϵ, δ)-*differentially private* if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all $X, Y \in \mathbb{N}^{|\mathcal{X}|}$ such that $||X - Y||_1 \leq 1$:

$$\Pr[\mathcal{M}(X) \in \mathcal{S}] \leq \exp(\epsilon) \Pr[\mathcal{M}(Y) \in \mathcal{S}] + \delta,$$

where the probability space is over the coin flips of the mechanism $\mathcal{M}$.
If $\delta = 0$, we say that $\mathcal{M}$ is ϵ-differentially private.[10]

Fact 10.5.
Some important facts about differential privacy:

- δ should be significantly smaller than the reciprocal of the size of the database to prevent a simple sampling approach from being classified as private.
- For any two neighboring databases (ϵ, δ)-differential privacy guarantees that the absolute privacy loss is bounded by ϵ with probability at least $1 - \delta$.
- No data-independent post-processing done to the result of a differentially private mechanism can decrease the privacy, i.e. the composition of any randomized data-independent mapping f and an (ϵ, δ)-differentially private mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private.
- Privacy loss is additive: The composition of an (ϵ_1, δ_1)-differentially private mechanism $\mathcal{M}_1$ and an (ϵ_2, δ_2)-differentially private mechanism $\mathcal{M}_2$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-differentially private.

We now want to consider the two most common techniques used to achieve differential privacy. The *Laplace mechanism* adds Laplacian noise to any output to hide the impact of a single individual. The *exponential mechanism* samples from the set of all outputs with a sampling distribution correlating with the utility of the outputs.

The Laplace Mechanism

The Laplace mechanism [23] is designed to provide privacy when numeric queries $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ are applied to databases $X \in \mathbb{N}^{|\mathcal{X}|}$. It achieves differential privacy by adding Laplacian noise to the output of a query. The amount of noise depends on the *sensitivity* of queries, which describes the magnitude of changes to the output when a single element is added to or removed from the database.

[10] An alternative model of differential privacy considers $||x||_1$ as public and compares databases of the same size that differ in a single element. Results using one of the definitions are usually adaptable to the other one as well [9].

Definition 10.6 (Sensitivity).
The ℓ_1-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is

$$\Delta f := \max_{\substack{X,Y \in \mathbb{N}^{|\mathcal{X}|} \\ ||X-Y||_1=1}} ||f(X) - f(Y)||_1.$$

Definition 10.7 (Laplace Distribution).
The *Laplace Distribution* centered at 0 with scale b is a distribution with probability density function

$$\text{Lap}(x \mid b) := \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$$

and variance $\sigma^2 = 2b^2$.

We write $\text{Lap}(b)$ to describe the Laplace distribution with scale b or a random variable $\Gamma \sim \text{Lap}(b)$.

Definition 10.8 (Laplace Mechanism [24]).
Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the *Laplace mechanism* is defined as

$$\mathcal{M}_L(X, f(\cdot), \epsilon) := f(X) + (\Gamma_1, \ldots, \Gamma_k)$$

where Γ_i are i.i.d. random variables drawn from $\text{Lap}(\Delta f/\epsilon)$.

A simple calculation shows:

Theorem 10.9 ([24]).
The Laplace mechanism preserves $(\epsilon, 0)$*-differential privacy.*

As the Laplace mechanism is the first and most prominent differential private mechanism, there exists a rich toolbox of results regarding composability, utility and applicability for a wide range of applications. See e.g. [24, 22, 73]. The most important property is additive composition as described in fact 10.5: When two or more queries are applied to the same database, privacy is diminished depending on the sum of all ϵ of the mechanisms involved. As a result any data collector must either accept a loss in privacy over time or instead limit the amount and sensitivity of queries allowed. While the first strategy is unacceptable in most applications, the second one might be achieved by releasing a fixed set of query results or instead decreasing ϵ after each query, ultimately consuming all utility of the data.

The Exponential Mechanism

There are situations in which the Laplace mechanism cannot be used effectively. For example, consider a multiset of elements that occur with different frequencies. A query might ask for the index of the element of highest frequency. Adding Laplacian noise to the correct output and potentially rounding to the next index does not result in a sensible result unless the elements are in order. However, applying the Laplace mechanism independently to queries of individual frequencies and reporting the highest value does not solve the problem either, as too much noise is required due to additive composability.

One solution to these problems is the *exponential mechanism* [53]. In the exponential mechanism all possible results of a query are evaluated according to their utility for a specific application. However, unlike ideal optimization algorithms there is no guarantee that the best solution is returned. Instead a probability distribution is defined on all possible outputs. By returning outputs of high utility with a higher probability than outputs of low utility, the returned data is guaranteed to be useful for the specific application without giving up privacy.

Before the exponential mechanism can be defined, a notion of utility sensitivity analogous to (range)-sensitivity defined in definition 10.6 is needed.

Definition 10.10 (Utility Sensitivity).
Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathcal{R}$, a *utility function* $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$ maps each possible result to a numeric utility. The *sensitivity* of a utility function $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$ is

$$\Delta u : \max_{r \in \mathcal{R}} \max_{\substack{X,Y \in \mathbb{N}^{|\mathcal{X}|} \\ ||X-Y||_1 = 1}} |u(X,r) - u(Y,r)|.$$

Definition 10.11 (Exponential Mechanism [24]).
The exponential mechanism $\mathcal{M}_E(X, u, \mathcal{R}, \epsilon)$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp\left(\frac{\epsilon \cdot u(X,r)}{2\Delta u}\right)$.

Theorem 10.12 ([24]).
The exponential mechanism preserves $(\epsilon, 0)$*-differential privacy.*

As the sensitivity is now defined as the difference in utility on neighboring databases, large differences in query results are no obstacle any more. Picking up the maximum frequency example from above it is conceivable that adding a single element to the multiset, the index of the element of largest frequency changes significantly. However, the utility of indices remains nearly identical as a single element cannot change the frequencies much. Hence, when only the index of the element of highest frequency is requested, the exponential mechanism is well better suited than the Laplace mechanism.

A strong argument for the use of the exponential mechanism are its utility guarantees. As the probability of returning outputs falls exponentially when they have worse utility, it is highly unlikely that an output with utility worse than the optimal utility minus $O((\Delta u/\epsilon) \log |\mathcal{R}|)$ is returned. (See [24] for details.)

Of course, the exponential mechanism also has some disadvantages. For example, the exponential mechanism has to evaluate utility for all or at least a significant amount of possible query outputs which results in an increase in time complexity compared to the Laplace mechanism.

10.2 Using Differential Privacy to Protect Data Release

Two trivial methods of publishing databases in a differentially private way are:

1. Query the frequency of every possible element from $\mathcal{X}$ and use the Laplace mechanism to protect the results. By combining and rounding all results a new database can be created and published. As the final aggregation is data-independent, the privacy is solely defined by the sensitivity of the queries.
2. Consider the set of all possible databases and a utility function u that measures similarity between the synthetic and original database in some way. By using the exponential mechanism a synthetic database similar to the original one is selected. For bigger ϵ the probability for a more similar database increases. Any operations and analyses on this database can be done without further reducing privacy.

Unfortunately, both approaches fail when attempting to implement them naively. While for the first approach, the number of queries is just too high to allow small ϵ when relying on additive composability, the second approach has two distinct problems. First, the set of all possible databases is tremendously large, preventing an efficient way to compute all utilities as needed. Further, there is no guarantee that the synthetic database is similar to the original one in any other way than measured by u. Hence, we have to break the elemental purpose of microaggregation: Release general purpose data without restrictions on specific use cases.

SmallDB

Over the years there have been several attempts to allow the differentially private publication of data by adapting one of the two attempts described above. The first algorithm we consider is called `SmallDB` [8, 9, 24]. It has been proposed by Blum et al. [8, 9] and extended by Dwork et al. [24] as an alternative to offline data synthesis using the exponential mechanism described in attempt 2 from above. While the disadvantage of restricted utility guarantees is still given, the time complexity and utility for specific queries of resulting data can be improved significantly by applying `SmallDB` instead of a naive implementation of the exponential mechanism. However, as we will see below, the algorithm is applicable only for large databases.

The utility guarantees of `SmallDB` are limited to so-called linear queries, which ask about the percentage or sum of elements with specific predicates g. In contrast to counting queries which allow only binary predicates $g : \mathcal{X} \to \{0, 1\}$, linear queries allow $g : \mathcal{X} \to [0, 1]$. For example, *How many smokers live in Vancouver?* is a linear query, while *What is the variance in dimension j of the database?* is not.

Definition 10.13 (Linear Query).
Given a universe $\mathcal{X} = \{\chi_1, \ldots, \chi_{|\mathcal{X}|}\}$ and a database $X \in \mathbb{N}^{|\mathcal{X}|}$, a *base query* $g : \mathcal{X} \to [0, 1]$ asks to what extend an element χ has a specific property.

Using a base query g, a *normalized linear query* $f : \mathbb{N}^{|\mathcal{X}|} \to [0, 1]$ has the form

$$f(X) := \frac{1}{||X||_1} \sum_{i=1}^{|\mathcal{X}|} x_i \cdot g(\chi_i).$$

Using a base query g, an *un-normalized linear query* $f : \mathbb{N}^{|\mathcal{X}|} \to [0, ||X||_1]$ has the form

$$f(X) := \sum_{i=1}^{|\mathcal{X}|} x_i \cdot g(\chi_i).$$

A *linear query* is either an un-normalized or a normalized linear query.

In SmallDB the range of the exponential mechanism is limited to databases of size $\frac{\log|\mathcal{Q}|}{\alpha^2}$ for a set of linear queries $\mathcal{Q}$ and an accuracy parameter $\alpha \in \mathbb{R}$. Further, the utility is measured as the negative of the maximum difference between the output of any query $f \in \mathcal{Q}$ on the original and synthetic database. See algorithm 24 for a detailed description of SmallDB.

Algorithm 24: SmallDB [24]

input : database $X \in \mathbb{N}^{|\mathcal{X}|}$, set of linear queries $\mathcal{Q}$, $\epsilon \in \mathbb{R}$ and $\alpha \in \mathbb{R}$
output: synthetic database $Y \in \mathbb{N}^{|\mathcal{X}|}$

1 Let $\mathcal{R} \leftarrow \{Y \in \mathbb{N}^{|\mathcal{X}|} : ||Y||_1 = \frac{\log|\mathcal{Q}|}{\alpha^2}\}$
2 Let $u : \mathbb{N}^{|\mathcal{X}|} \times \mathcal{R} \to \mathbb{R}$ with $u(X, Y) := -\max_{f\in\mathcal{Q}} |f(X) - f(Y)|$
3 Sample and Output $Y \in \mathcal{R}$ with $\mathcal{M}_E(X, u, \mathcal{R}, \epsilon)$

$(\epsilon, 0)$-differential privacy of SmallDB follows directly from the $(\epsilon, 0)$-differential privacy of the exponential mechanism $\mathcal{M}_E(X, u, \mathcal{R}, \epsilon)$. It should be noted that SmallDB is indeed able to achieve sufficiently low time consumption even for very large databases, as the size of $\mathcal{R}$ depends on $|\mathcal{Q}|$ and α alone. Using the fact $\Delta u \leq 1/||X||_1$ for normalized linear queries, SmallDB achieves the following utility guarantee for queries included in $\mathcal{Q}$:

Theorem 10.14 ([24]).
Let $\mathcal{Q}$ be any class of normalized linear queries. Let Y be the database output by SmallDB$(X, \mathcal{Q}, \epsilon, \alpha)$. *Then with probability $1 - \beta$:*

$$\max_{f\in\mathcal{Q}} |f(X) - f(Y)| \leq \alpha + \frac{2\left(\frac{\log|\mathcal{X}|\log|\mathcal{Q}|}{\alpha^2} + \log\left(\frac{1}{\beta}\right)\right)}{\epsilon||X||_1}.$$

Further, for any database x with

$$||X||_1 \geq \frac{16\log|\mathcal{X}|\log|\mathcal{Q}| + 4\log\left(\frac{1}{\beta}\right)}{\epsilon\alpha^3}$$

SmallDB$(X, \mathcal{Q}, \epsilon, \frac{\alpha}{2})$ *achieves with probability $1 - \beta$*

$$\max_{f\in\mathcal{Q}} |f(X) - f(Y)| \leq \alpha.$$

As can be seen in algorithm 24 and theorem 10.14 neither complexity nor privacy of SmallDB depends on the size of the database. However, utility does. For a worst case

precision loss of α to hold with high probability, theorem 10.14 requires a minimum database size in relation to the size of $\mathcal{Q}$. Solving the size constraint for $|\mathcal{Q}|$ we obtain

$$|\mathcal{Q}| \leq \sqrt[4\log(|\mathcal{X}|)]{\beta} \exp\left(\frac{\alpha^3 \epsilon ||X||_1}{16 \log(|\mathcal{X}|)}\right). \tag{10.1}$$

To allow the set of queries needed to reconstruct properties of X to be large enough, equation 10.1 dictates that the accuracy factor α as well as the privacy leakage ϵ and the size of the database $||X||_1$ must be large in relation to the size of the set of possible database elements. Let us consider a simple example to analyze whether this bound is enough to allow a publication of databases similar to those typically handled by microaggregation. Consider a discretization of 5-dimensional data into 20 different values in each dimension, i.e. $|\mathcal{X}| = 20^5$. Further assume $\epsilon = 0.1$ and $\beta = 0.01$. In table 10.1 the resulting upper limits on $|\mathcal{Q}|$ according to equation 10.1 are shown. As can be seen, although we use a generously rough discretization of a low-dimensional element domain and allow large errors α (remember, we consider normalized linear queries ranged between 0 and 1) `SmallDB` seems to be unable to handle reasonably small databases based on the guarantees given by Dwork et al. Further relaxing ϵ and β does not help either as their effect in equation 10.1 is very limited.

Table 10.1: Number of queries answerable for $\epsilon = 0.1, \beta = 0.01$ and different values for α and $||X||_1$. $|\mathcal{X}|$ has been set to 20^5.

α	$\|\|X\|\|_1$	$\max \|\mathcal{Q}\|$
10^{-2}	10^3	0
	10^6	0
	10^7	0
	10^8	0
	10^9	1
10^{-1}	10^3	0
	10^6	1
	10^7	60
	10^8	10^{18}
	10^9	10^{181}

Further decreasing $|\mathcal{X}|$ does help, but becomes more and more unrealistic for large databases of millions of elements. Although the number of queries is allowed to be exponential in $||X||_1$ this result does not translate well into the context of data release. As we cannot assume strict constraints on the use-case for the resulting database, it would be a stretch to assume that a sub-exponential number of queries would be enough to accurately translate the original database X into a synthetic database Y.

It should be obvious from the example that theorem 10.14 is not enough to allow `SmallDB` in the context of general-purpose private data release as targeted by microaggregation. However, by restricting queries to counting queries (i.e. restricting base queries to $g : \mathcal{X} \to \{0, 1\}$) and using results from learning theory, Blum et al. were able to provide stronger guarantees. To quote the stronger bounds we first need to introduce the concepts of *shattering* and *VC dimension* from the field of algorithmic learning theory. Again, the definitions are adapted from [24] to assure notational consistency.

Definition 10.15 (Shattering).
A class of counting queries $\mathcal{Q}$ *shatters* a collection of elements $S \subseteq \mathcal{X}$ if for every $T \subseteq S$, there exists an $f \in \mathcal{Q}$ such that $\{s \in S : f(s) = 1\} = T$.

Definition 10.16 (Vapnik-Chervonenkis (VC) Dimension).
A collection of counting queries $\mathcal{Q}$ has *VC-dimension* d if there exists some set $S \in \mathcal{X}$ of cardinality $|S| = d$ such that $\mathcal{Q}$ shatters S, and $\mathcal{Q}$ does not shatter any set of cardinality $d + 1$. This quantity is denoted by *VC-DIM*($\mathcal{Q}$).

Informally speaking, if a set of queries $\mathcal{Q}$ shatters a set S, every subset T of S is uniquely defined by a query $f \in \mathcal{Q}$. Further, if for example *VC-DIM*($\mathcal{Q}$) = 3, there is no subset of four elements from $\mathcal{X}$ such that each selection of these four elements is uniquely defined by a function $f \in \mathcal{Q}$, whereas there is at least one such subset of size 3. Typically by the VC dimension the expressiveness of a set of queries is measured. Query sets with larger VC dimension can express more complex questions but are harder to learn from examples.

Equipped with the notion of VC dimension we are now able to consider a stronger utility guarantee for `SmallDB` than that offered by theorem 10.14.

Theorem 10.17 ([24]).
Let $\mathcal{Q}$ be any class of normalized counting queries. For any database X with

$$||X||_1 \geq O\left(\frac{\log|\mathcal{X}|\,\textit{VC-DIM}(\mathcal{Q}) + \log\left(\frac{1}{\beta}\right)}{\epsilon\alpha^3}\right)$$

$\texttt{SmallDB}(X, \mathcal{Q}, \epsilon, \frac{\alpha}{2}) =: Y$ *achieves with probability* $1 - \beta$

$$\max_{f\in\mathcal{Q}} |f(X) - f(Y)| \leq \alpha.$$

The new result is stronger than the previous one, because for any finite set of queries $\mathcal{Q}$ it holds *VC-DIM*($\mathcal{Q}$) $\leq \log|\mathcal{Q}|$. (See [24].) Analogous to equation 10.1 we can reformulate theorem 10.17 into an upper bound for *VC-DIM*($\mathcal{Q}$):

$$\textit{VC-DIM}(\mathcal{Q}) \leq O\left(\frac{\alpha^3\epsilon||X||_1 - \log\left(\frac{1}{\beta}\right)}{\log(|\mathcal{X}|)}\right) \tag{10.2}$$

Ignoring constants we can use equation 10.2 to see whether the applicability of `SmallDB` has improved compared to the previous example using equation 10.1. The result can be found in table 10.2.

To evaluate the usefulness of `SmallDB` for data release we need to determine the VC dimension of reconstruction queries. As we are restricted to normalized counting queries and do not assume which characteristics of the data are interesting to a data analyst, an obvious choice would be range queries. By dividing the domain into small sub-spaces and asking for the relative frequency of elements within these sub-spaces we obtain data usable to construct a neutral representation of the original database, given the sub-domains are small enough. Theorem 10.18 shows that the VC dimension of such queries is $2d$ for d-dimensional data and sub-spaces defined by axis-aligned $d - 1$ dimensional hyperplanes. While the result seems to be of general interest, I could not find an existing proof allowing arbitrary values of d. Hence, a newly designed proof is included below.

Table 10.2: Rough estimate on the maximum VC dimension of a query set answerable for $\epsilon = 0.1, \beta = 0.01$ and different values for α and $||X||_1$. $|\mathcal{X}|$ has been set to 20^5.

α	$\|\|X\|\|_1$	max *VC-DIM*($\mathcal{Q}$)
10^{-2}	10^3	0
	10^6	0
	10^7	0
	10^8	0
	10^9	6
10^{-1}	10^3	0
	10^6	6
	10^7	66
	10^8	667
	10^9	6675

Theorem 10.18.
Let $\mathcal{Q}_d$ for $d \in \mathbb{N}$ be a set of normalized range queries $f : \mathbb{N}^{|\mathcal{X}|} \to [0, 1]$, $\mathcal{X} \subseteq \mathbb{R}^d$. Each query f counts elements within a connected sub-space $\mathcal{S}_f \subseteq \mathbb{R}^d$ defined by $d-1$ dimensional hyperplanes that each are perpendicular to one of the unit vectors. VC-DIM($\mathcal{Q}_d$) $= 2d$.

Proof. We first need to show that for arbitrary values of $d \in \mathbb{N}$ the set of queries $\mathcal{Q}_d$ defined by the sub-spaces shatters a set of elements $S_d \subseteq \mathbb{R}^d$ with $|S_d| = 2d$.

The set S_d is defined as the set of all negated and non-negated d-dimensional unit vectors, i.e. $S_d = \{(1, 0, \ldots, 0), (-1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1), (0, \ldots, 0, -1)\}$. For $d = 1$, $S_1 = \{1, -1\}$. It should be obvious that $\mathcal{Q}_1$ shatters S_1. It remains to show that S_d is indeed shattered by $\mathcal{Q}_d$ for $d \geq 2$. By induction we know that $\mathcal{Q}_{d-1}$ shatters S_{d-1}. Each element of S_d is 0 in every but one dimension. For each dimension j there are exactly two elements s_1 and s_2 with $s_1^j = 1$ and $s_2^j = -1$. By considering intervals $[-0.5, 0.5]$, $[-1, 0.5]$, $[-0.5, 1]$ and $[-1, 1]$ in dimension j we can create sub-spaces including each subset of $\{s_1, s_2\}$ together with all other elements in S_d. Combinations of one interval per dimension result in a set of sub-spaces of $\mathbb{R}^d$ including a set T_d for each subset of S_d. The set of queries $\mathcal{Q}'_d \subseteq \mathcal{Q}_d$ corresponding to these sub-spaces shatters S_d. So S_d is shattered by $\mathcal{Q}_d$ as well. We obtain *VC-DIM*($\mathcal{Q}_d$) $\geq 2d$.

We now show *VC-DIM*($\mathcal{Q}_d$) $\leq 2d$. Let $f \in \mathcal{Q}_d$ be the query corresponding to the smallest sub-space $\mathcal{S}$ including all elements from a set S_d of size $2d$. In the first part of this proof we have seen that a set of elements S_d can be shattered when each of the elements lies on exactly one hyperplane bounding $\mathcal{S}$. We now want to show that this is a necessary property for any set of elements S_d that can be shattered by $\mathcal{Q}_d$. It should be obvious that no element s can lie inside $\mathcal{S}$ but not on one of its bounding hyperplanes, as otherwise we could not separate s from $S_d \setminus \{s\}$ by any query $f' \in \mathcal{Q}_d$. It remains to show that no two elements s and s' share a common bounding hyperplane or in other words: each element from S_d is the unique smallest or biggest element in one of the dimensions, when S_d is projected onto this dimension.

Consider a bipartite graph $G = (V, W, E)$. Each node $v_i \in V$ corresponds to an element $s_i \in S_d$, each node $w_j \in W$ corresponds to one of the bounding hyperplanes p_j of $\mathcal{S}$. The edge $(v_i, w_j) \in E$ indicates that the element s_i corresponding to v_i lies on the hyperplane

p_j corresponding to w_j. As we have seen above, each element lies on at least one of the bounding hyperplanes, so there is no isolated node within V.

We now want to label nodes from G to describe a subset of $T \subseteq S_d$ that can be separated from $S_d \backslash T$ by a valid query $f' \in \mathcal{Q}_d$. For this we use a labeling function $l : V \cup W \to \{+, -\}$. By labeling a node $v_i \in V$ *positive*, i.e. $l(v_i) = +$, we indicate that $s_i \in T$, analogously a *negative* label at a node $v_{i'} \in V$ indicates $s_{i'} \notin T$. Labels of nodes $w_j \in W$ are determined by the labels of nodes $v_i \in V$ connected to them by edges $e \in E$. When $l(v_i) = +$, s_i should be included in T. Hence, we cannot shrink $\mathcal{S}$ by moving p_j towards the center of $\mathcal{S}$. This stability property of p_j is indicated by labeling w_j positive. However, when no node v_i connected to a node w_j is labeled positive, w_j is labeled negative.

To see that in S_d no two elements can share a common bounding hyperplane of $\mathcal{S}$, we assume any order of elements $s_1, \ldots, s_{2d}$. To construct a contradiction assume s_1 lies on exactly two hyperplanes. We now begin to label G from v_1 on to v_{2d} with positive labels. A visualization of the labeling process is given as figure 10.3. After labeling v_1 positive, the number of positive labels in V is 1 and in W is 2. Now, node after node, elements and hyperplanes are labeled positive. The core observation of this construction is that a node v_i cannot exclusively be connected to nodes w_j, which are already labeled positive, as this would translate to the inseparability of s_i from $s_1, \ldots, s_{i-1}$. Hence, the labeling of each element leads to the labeling of at least one additional hyperplane. So at some point throughout the labeling process, every element w_j is labeled positive. Most importantly, this happens before node v_{2d} is labeled. As v_{2d} cannot be connected to any non-positively labeled node w_j, a contradiction is obtained: It is impossible to shatter S_d, when s_1 is part of it. Hence, elements cannot lie on the intersection of two hyperplanes bounding $\mathcal{S}$. As there are $2d$ elements and each of them must lie on exactly one bounding hyperplane, $\mathcal{S}$ can only be minimal when there are no two elements sharing a hyperplane.

It remains to be shown that no set S'_d of size $2d + 1$ can be shattered by $\mathcal{Q}_d$. Note that any subset S of a set S' which is shattered by a set of queries $\mathcal{Q}$ is also shattered by $\mathcal{Q}$. Now consider the extension of a set S_d with $|S_d| = 2d$ by an additional element s'. Let $f \in \mathcal{Q}_d$ be the query corresponding to the smallest sub-space including all elements from S_d. As we have shown above, each of the $2d$ hyperplanes bounding the sub-space defined by f has a single element $s \in S_d$ on it. Hence, there can be no hyperplane on which s', but no element from s lies on. However, as we have seen above, two elements cannot share a common hyperplane when there are at least $2d$ elements. Further, for the same reason as discussed above, s' cannot lie inside the sub-space but not on one of the hyperplanes. There is only one case left: s' is outside of the sub-space defined by f, i.e. there is one dimension j for which s' is smaller or larger than any element from S_d. This implies that now there is an element $s \in S_d$ inside the smallest sub-space defined by a query f' including $S_d \cup \{s'\}$ that does not lie on one of the bounding hyperplanes. Hence, there is no query $f'' \in \mathcal{Q}_d$ which includes $S_d \setminus \{s\}$ and s', but not s.

As none of the options lead to a set that is shattered by $\mathcal{Q}_d$, we conclude that no set S_d of $2d$ elements can be extended by an additional element without losing the shatterability property. So, *VC-DIM*$(\mathcal{Q}_d) \leq 2d$.

□

Before we move on to the next approach, let us summarize the results of `SmallDB`. By

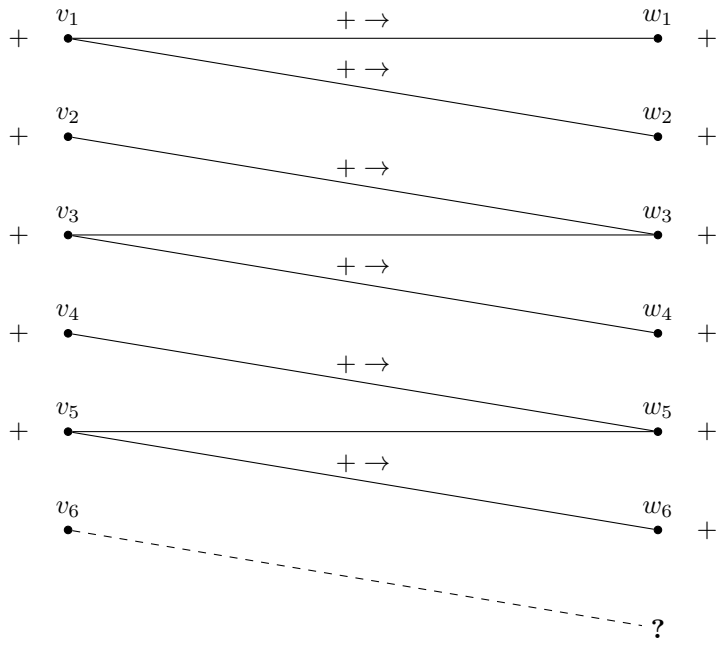


Figure 10.3: Example graph relating elements $s \in S_d$ and bounding hyperplanes of subspace $\mathcal{S}$. As s_1 is located on the intersection of two hyperplanes p_1 and p_2, there is no valid labeling of nodes. Edge labels indicate which v_i cause the $w_j \in W$ to be labeled positive.

limiting the range of the exponential mechanism to databases of sizes defined only by $|\mathcal{Q}|$ and α we are able to execute it more efficiently. However, only very large databases over small data domains as inputs allow for small accuracy parameters α. Re-evaluating the example shown in table 10.2 in the context of range queries on d-dimensional data shows that more than a million elements are needed to handle coarsely discretized data for large error terms α. Hence, the applicability of `SmallDB` is quite limited even considering queries with limited VC dimension.

Interval Queries

As we have seen above, it may not be possible to reconstruct small databases using the exponential mechanism. Another approach called `ReleaseIntervals` published by Blum et al. ([8, 9]) adapts trivial strategy 1, i.e. applies the Laplace mechanism directly, to reconstruct one-dimensional databases with interval queries. Although the original approach is described for one-dimensional data only, it could be adapted to higher-dimensional data, as we will discuss below.

Normalized interval queries are the one-dimensional equivalent of normalized range queries. Given a database domain $\mathcal{X} \subseteq \mathbb{R}$ and two elements $a_1 \in \mathcal{X}$ and $a_2 \in \mathbb{R}$ a normalized interval query $f_{a_1,a_2} : \mathbb{N}^{|\mathcal{X}|} \to [0,1]$ is a normalized counting query asking for the relative frequency of elements from a database X between a_1 and a_2. As normalized interval queries are normalized counting queries, their sensitivity is $\Delta f = \frac{1}{||X||_1}$ which dictates the amount of Laplacian noise needed to protect them from leaking private information.

In `ReleaseIntervals` intervals instead of possible elements are queried so the trade-off

between privacy, utility and performance can be adjusted and the privacy budget ϵ can be used more targeted. Whereas trivial strategy 1 uses large parts of its privacy budget to protect individual possible elements that are not even part of the database, which requires a coarse discretization and a large time budget, the protection of interval queries is more resource-conserving.

`ReleaseIntervals` uses a data-dependent adjustment of interval sizes to reflect different data densities in different regions of the data domain without assuming particular distributions. Interval sizes are adjusted to contain about $\frac{\alpha}{6 \cdot ||X||_1}$ elements. As this data-dependent segmentation is based on binary search and uses the same interval queries used to create the output, little additional privacy budget is allocated to protect data-dependent interval selection. After `ReleaseIntervals` has determined intervals, a multiple of $\frac{\alpha}{6 \cdot ||X||_1}$ elements are randomly drawn from $\mathcal{X}$ within the borders of each interval. See algorithm 25 for a detailed description of `ReleaseIntervals`. Privacy and utility guarantees for `ReleaseIntervals` are shown in theorem 10.19 and theorem 10.20, respectively.

Algorithm 25: `ReleaseIntervals` [9]

input : database $X \in \mathbb{N}^{|\mathcal{X}|}$ for $\mathcal{X} = \{1, \ldots, 2^\sigma\}$, $\epsilon \in \mathbb{R}$ and $\alpha \in \mathbb{R}$
output: synthetic database $Y \in \mathbb{N}^{|\mathcal{X}|}$

Let $\alpha' \leftarrow \alpha/6$
Let $MaxIntervals \leftarrow \lceil 4/3\alpha' \rceil$
Let $\epsilon' \leftarrow \epsilon/(\sigma \cdot MaxIntervals)$
Let $Bounds$ be an array of length $MaxIntervals$
Let $i \leftarrow 1$
Let $Bounds[0] \leftarrow 1$
while $Bounds[i-1] < 2^\sigma$ **do**
    Let $a \leftarrow Bounds[i-1]$
    Let $b \leftarrow (2^\sigma - a + 1)/2$
    Let $inc \leftarrow (2^\sigma - a + 1)/4$
    while $inc \geq 1$ **do**
        Let $v \leftarrow f_{a,b}(X) + \mathrm{Lap}(1/(\epsilon'||X||_1))$
        if $v < \alpha'$ **then**
            $b \rightarrow b - inc$
        else
            $b \rightarrow b + inc$
        $inc \leftarrow inc/2$
    Let $Bounds[i] \leftarrow b$
    $i \leftarrow i + 1$
Output Y, a database that has $\alpha' m$ elements in each interval $[Bounds[j-1], Bounds[j]]$ for each $j \in [i]$, for any $m > 1/\alpha'$.

Theorem 10.19 ([9]).
For any $\epsilon \in \mathbb{R}$, $\alpha \in \mathbb{R}$ and $X \in \mathbb{N}^{|\mathcal{X}|}$, `ReleaseIntervals`$(X, \epsilon, \alpha)$ preserves $(\epsilon, 0)$-differential privacy.

Theorem 10.20 ([9]).
Let $\mathcal{Q}$ be a set of normalized interval queries. For $\epsilon, \alpha, \beta \in \mathbb{R}$ and any database $X \in \mathbb{N}^{|\mathcal{X}|}$ with $\mathcal{X} = \{1, \ldots, 2^\sigma\}$ and

$$||X||_1 \geq \frac{288\sigma}{\epsilon\alpha^3} \cdot \log\left(\frac{8\sigma}{\beta\alpha}\right)$$

`ReleaseIntervals`$(X, \epsilon, \alpha) =: Y$ *with probability $1 - \beta$ achieves*

$$\max_{f \in \mathcal{Q}} |f(X) - f(Y)| \leq \alpha.$$

Necessary database sizes for reasonable values of ϵ, α, β and σ are shown in table 10.4. As can be seen, while the size requirement grows only slowly with the size of the database domain, even for small domains of $2^\sigma = 2^5 = 32$ elements, at least 10^8 elements are needed to achieve even weak utility guarantees of $\alpha = 0.1$ and $\beta = 0.01$.

Table 10.4: Minimum database sizes for utility guarantee to hold. As in previous examples, $\epsilon = 0.1$ and $\beta = 0.01$ have been used. Different values for α and σ reflect different accuracy requirements and domain sizes.

α	σ	$\min \|\|X\|\|_1$
10^{-2}	5	10^{11}
	10	10^{11}
	50	10^{12}
	100	10^{12}
10^{-1}	5	10^8
	10	10^8
	50	10^9
	100	10^9

As claimed by Blum et al. in [9], `ReleaseIntervals` can be adapted to higher dimensional data. For this, fix a dimension j and execute standard `ReleaseIntervals` up until after step 19 of algorithm 25. Afterwards, each interval serves as a new database to be handled individually by applying `ReleaseIntervals` with adjusted ϵ and α on another dimension j'. The resulting process is similar to `MONDRIAN` discussed in chapter 7. However, dimensions are handled in order and only once but are segmented in more smaller chunks at once.

Privacy of the resulting algorithm is $d \cdot \epsilon$ which can be explained using additive composition of the Laplace mechanism and the fact that the number of interval queries needed is now multiplied by d. Generalizing theorem 10.20 to show utility bounds is not that simple as numerous intervals used to create elements of the synthetic database can now influence the total error of a range query. In one dimension there can only be two intervals created by `ReleaseIntervals` that partly overlap with any given interval $[a, b]$. However, answering an arbitrary range query in more than one dimension might involve data from all ranges computed by `ReleaseIntervals` as a sub-space is not limited by two points but by higher dimensional hyperplanes. E.g. for two-dimensional data, sub-spaces are limited by axis-parallel edges, which can intersect with an arbitrary number of ranges used to define Y.

Histogram Queries

The goal of previous approaches presented in this section has been to create synthetic databases that are in some way similar to the input database but aggregated and hence smaller. We now take a look at a different approach. Instead of querying the database and using the data to build a new smaller one, we publish the query results itself to create a new database of the same size of the original one. A simple mechanism called `LaplaceHistograms` presented in [73] is able to compute a noisy histogram of the data while allowing small databases, errors and privacy parameters ϵ. It is essentially a precise formalization of the first trivial method presented in the beginning of this section. However, by avoiding additive composability it is sufficient to add relatively little noise.

The question *How many elements of type χ are there?* is formalized by so-called *histogram queries*:

Definition 10.21 (Histogram Query).
Given a universe $\mathcal{X} = \{\chi_1, \ldots, \chi_{|\mathcal{X}|}\}$ and a database $X \in \mathbb{N}^{|\mathcal{X}|}$ a *normalized histogram query* $f_\chi : \mathbb{N}^{|\mathcal{X}|} \to [0,1]$ for $f_{\chi_i}(X) := \frac{1}{||X||_1} x_i$ is a normalized linear query counting the number of occurrences of an element χ_i within a database X.

Algorithm 26: `LaplaceHistograms` [73]

input : database $X \in \mathbb{N}^{|\mathcal{X}|}$ drawn from any universe $\mathcal{X}$, $\epsilon \in \mathbb{R}$
output: noisy histogram $H \in \mathbb{R}^{|\mathcal{X}|}$ of X

foreach $x_i \in X$ **do**
  Let $h_i = f_{\chi_i}(X) + \text{Lap}(\frac{1}{\epsilon \cdot |\mathcal{X}|})$
Let $H = (h_1, \ldots, h_{|\mathcal{X}|})$

As all but one query f_χ for all $\chi \in \mathcal{X}$ deliver the same results on neighboring databases, the sensitivity of a combined query asking for the normalized frequency vector of every possible universe element is $1/|\mathcal{X}|$ instead of $n \cdot 1/|\mathcal{X}|$ for n arbitrary normalized counting queries. As a result, Laplacian noise of scale $1/(\epsilon \cdot |\mathcal{X}|)$ is enough to guarantee ϵ differential privacy:

Theorem 10.22 ([73]).
For any $\epsilon \in \mathcal{R}$ and $X \in \mathbb{N}^{|\mathcal{X}|}$, `LaplaceHistograms`$(X, \epsilon)$ preserves $(\epsilon, 0)$-differential privacy.[11]

Proof. On neighboring databases X and Y only a single element is omitted or added. Hence, X and Y differ on only one position i by either 0, +1 or −1, i.e. $\Delta H = 1/|\mathcal{X}|$. According to the privacy guarantee of the Laplace mechanism (theorem 10.9) `LaplaceHistograms`(X, ϵ) preserves $(\epsilon, 0)$-differential privacy. □

Compared to `SmallDB` and `ReleaseIntervals`, `LaplaceHistograms` has some important advantages. First of all, it is applicable to small and large databases alike. Further,

[11] In the original paper a different definition of differential privacy has been used. The statement and proof have been adapted to fit the notation presented in definition 10.4.

there is no restriction on dimensionality or a minimum database size to guarantee utility. Its most important downside is probably its resource consumption. As $|\mathcal{X}|$ directly effects the number of counting queries and the number of independent random noise values to be generated, a coarse discretization of the data domain is needed to keep performance cost manageable.

Strongly-Safe k-Anonymization

In definition 3.16 we presented strongly safe k-anonymization introduced by Li et al. [40] as an adaption of the standard k-anonymity guarantee. Now that some basics of differential privacy have been defined, we can re-evaluate this technique to see what it offers in the context of differential privacy.

Recall that strongly safe k-anonymization is a three-step process defined by a data-independent clustering $\mathcal{C}^{ind}$ followed by the collection of cluster statistics of each cluster when $\mathcal{C}^{ind}$ is applied to a database X and a final step in which results from clusters of $k-1$ or less elements are removed to achieve k-anonymity.

As the first step is data-independent, no privacy loss occurs in defining the clusters. Nevertheless, as Li et al. show, strongly safe k-anonymization does not offer (ϵ, δ)-differential privacy for any $\delta < 1$ unless only a single cluster containing all elements is used in step 1.

The reason for this observation is simple: When the database domain $\mathcal{X}$ is divided into more than one cluster, e.g. into two clusters C_1 and C_2, there must exist neighboring databases X_1 and X_2, so that the amount of elements in C_1 and C_2 differs between X_1 and X_2. Hence, even simple queries asking for the number of elements within each cluster would reveal with certainty, whether X_1 or X_2 has been clustered. Expressing the core argument of this proof in other words results in the following well-known fact:

Remark 10.23.
An (ϵ, δ)-differentially private mechanism cannot be deterministic, except $\delta = 1$.

Li et al. further evaluated how pre-processing in the form of random sampling can be used in combination with strongly safe k-anonymization to achieve differential privacy. Note that as a result a private mechanism that does not rely on Laplacian noise or the exponential mechanism is obtained.

Theorem 10.24 ([40]).
Let $f(j; n, \beta)$ denote the probability of getting exactly j successes in n trials where each trial succeeds with probability β, i.e. $f(j; n, \beta)$ is the probability mass function for the binomial distribution. Let $k \in \mathbb{N}$, $0 < \beta < 1$, $\epsilon \geq -\log(1-\beta)$, $\gamma = \frac{\exp(\epsilon)-1+\beta}{\exp(\epsilon)}$ and

$$\delta := \max_{n: n \geq \lceil \frac{k}{\gamma} - 1 \rceil} \sum_{j > \gamma n}^{n} f(j; n, \beta).$$

Any strongly safe k-anonymization algorithm is (ϵ, δ)-differentially private when preceded by a random sampling in which any element from a database X is included with probability β.

As the privacy result given in theorem 10.24 is rather hard to grasp, consider some examples of valid combinations of β, ϵ and δ shown in table 10.5. While privacy seems

to be no issue according to table 10.5, there are no utility guarantees as for the other mechanisms described above. The most hard to quantify type of utility loss is caused by the data-independent clustering step. Without assuming the data follows a known distribution, there is no way to bound utility loss to reasonable levels. Keep in mind that clusters that contain $k-1$ or less elements have to be removed and there is further no mechanism to prevent a large fraction of elements to be contained in a single cluster.

Table 10.5: Combinations of β, ϵ and δ that guarantee (ϵ, δ)-differential privacy for strongly safe k-anonymization preceded by β-sampling for $k = 20$. Data from [40].

β	ϵ	δ
0.05	0.25	$7 \cdot 10^{-10}$
	0.5	$4 \cdot 10^{-6}$
	1	$2 \cdot 10^{-3}$
0.1	0.25	$3 \cdot 10^{-14}$
	0.5	$2 \cdot 10^{-9}$
	1	$8 \cdot 10^{-6}$
0.2	0.25	$2 \cdot 10^{-19}$
	0.5	$4 \cdot 10^{-14}$
	1	$6 \cdot 10^{-9}$

A possible remedy would be to replace the data independent clustering by a clustering algorithm that is $(\epsilon, 0)$-differentially private which leads to the notion of ϵ-safe k-anonymity for which similar results to theorem 10.24 can be shown. (See [40] for details.) In summary, strongly-safe and ϵ-safe k-anonymization offer a way to publish cluster statistics without assuming analysis goals, similar to microaggregation. It further shows that creating an $(\epsilon, 0)$-differentially private clustering algorithm suffices to build a private data release mechanism that does not require very large database sizes. In the next section we take a look at a possible approach to create such clusterings by adapting the `MONDRIAN` algorithm.

10.3 New Approaches to Differentially Private Data Release

As we have seen in the previous section, there are many approaches to combine differential privacy with the release of data. However, due to problems in regard of utility and complexity, especially for small databases and large data domains, so far there is no algorithm able to replace microaggregation entirely. Algorithms like `SmallDB` and `ReleaseIntervals` offer complete solutions using standard techniques but good utility can be achieved only on large and/or one-dimensional databases unless severe compromises are made regarding privacy. Frameworks like safe k-anonymization allow the transformation of k-member clustering algorithms into differentially private data release mechanisms. As the major part of utility loss using this framework is due to the clustering algorithm, there might be opportunity in moving forward by designing clustering mechanisms that create stable clusters on neighboring databases.

Unfortunately, maximum distance and Lloyd-based microaggregation heuristics as described in chapter 5 and 6 are particularly unstable, when it comes to small data perturbations. A minimum requirement to achieve reasonable privacy in the sense of (ϵ, δ)-differential privacy is that clusters stay mostly intact with only a few elements switching clusters, when a single element is missing form the input database. `MONDRIAN` and its variants described in section 7.1 seem to be better suited for this task.

Adapting Mondrian

Recall that in `MONDRIAN` there are two alternating tasks and a break condition that define a k-member clustering: First, we need to find a good splitting dimension that defines on which axis or between which axes to split. Next, we need to decide where to split exactly. In all previous incarnations of `MONDRIAN` we used the median of the data projected onto the splitting dimension as the split position. However, to achieve privacy, we need to re-evaluate whether it might be a good idea to move the split slightly or to do fuzzy splitting in a way that elements closer to the splitting hyperplane are more likely to switch sides. Finally, when splitting results in smaller and smaller databases we need to re-consider the break condition, because a hard size limit of $2k - 1$ as used e.g. in `MONDRIAN_V` is easily detectable. While I did not complete a full algorithm during my research, I want to present some interesting insights and approaches.

Computing Variances

One core insight learned from `MONDRIAN_V` is that the selection of split dimensions based on maximum variances is better suited to guarantee high cluster homogeneity than traditional splitting rules based on maximum diameter measures.

A good starting point would be to look at differentially private variance computations. A recent paper by Du et al. discusses this topic in detail [21]. The authors describe a tool called `EXPQ` which uses the exponential mechanism to estimate arbitrary quantiles of single dimensional data. By using `EXPQ` in the way described below, the authors are able to compute an estimation of median and standard deviation. Firstly, an estimation m of the median is computed by using `EXPQ` to find the center quantile of the data. Afterwards a new database X' is created by taking Euclidean distances between all data elements and m. In a final step `EXPQ` is used to estimate the center quantile of X' which is also an estimation for the standard derivation of X. The result is ϵ-differentially private if the sum of privacy budgets for both `EXPQ` calls is ϵ.

By using this approach we obtain an estimation of standard derivation instead of variance. However, as we are only interested in which dimension has the highest variance, there is no need to increase the error by squaring the result obtained. Note that the algorithm uses the median instead of the mean to compute distances to the data center. While this approach increases errors, the median is more stable than the mean, which allows for a good utility-privacy trade-off.

Avoiding Additive Composition

To find the dimension of highest variance of a database consisting of d-dimensional data, we can use the procedure described above on each dimension individually and take the one that reports the highest value. However, due to additive composition of these variance queries we can only use $\epsilon' = \epsilon/d$ for each of the queries to obtain ϵ-differential privacy for the mechanism of finding the dimension of maximum variance. This result seems to be sub-optimal because we pay with our privacy budget for a lot of information that is not needed. Unlike for histogram queries described in the previous section, the omission or addition of a single element changes the output of all queries, as a single element influences variances in every dimension.

As only the index of the dimension of largest variance is needed, one might hope to reduce privacy loss by avoiding to output individual variances as intermediate results. In [24] a technique called *Report Noisy Max* is introduced to avoid additive composition for counting queries, in which each database element potentially influences all counts. Report Noisy Max uses Laplacian noise of scale $1/\epsilon$ added to each of the d query results and guarantees ϵ differential privacy under the condition, that only the index of the noisy query of highest value is returned. Hence, compared to a naive approach, a factor d of additional noise scale is saved.

Unfortunately, the privacy proof of Report Noisy Max relies heavily on the fact that the Laplacian mechanism and counting queries are used. However, as `EXPQ` uses the exponential mechanism and variance or standard deviation might change significantly, when single elements are removed, Noisy Arg Max cannot be used without adaptation.

Splitting and Breaking

Beside the problem of finding a good splitting dimension, we have to protect the actual splitting process. For this we can rely on `EXPQ` to find a noisy median value of the data projected onto the splitting dimension. Splitting accordingly and repeating the process over and over leads to an ϵ-differentially private clustering process as requested by ϵ-safe k-anonymization. Hence, preceding the clustering process by a random sampling and only publishing cluster centroids and cluster sizes should allow privacy guarantees.

There is, however, another problem. For standard k-anonymity we stopped the splitting, when otherwise clusters with less than k elements would be created. Again, when considering differential privacy, such hard bound cannot be obtained because it would be detectable. However, when we drop the strict size constraint and stop the process at a noisy threshold like $\alpha k + \mathrm{Lap}(1/\epsilon)$ for suitable α and ϵ, we can still obtain safe k-anonymization without the necessity of cluster deletion.

10.4 Summary

In this chapter we have seen the possibilities and problems of applying differential privacy to protect the release of sensitive information. Although there are several algorithms trying to achieve this goal and despite promising attempts on a way to adapt the `MONDRIAN` algorithm,

there still is no solution that replaces microaggregation in the context described throughout this thesis. It remains to state that differential privacy is indeed the way to go, when attacker knowledge is unpredictable and privacy is the highest priority when releasing data. For all other cases involving the release of sensitive data, a modern microaggregation heuristic like ONA* which is more efficient and utilitiy-preserving should be considered even though privacy guarantees are weaker.

11

Conclusion

In this thesis we have discussed and advanced many aspects of k-anonymous microaggregation. Before we come to a close let us recap the results of this dissertation and discuss open problems and directions for further advancements regarding this topic. The first main part of this thesis is the development and evaluation of several new heuristics, lowering the time complexity and distortion cost of microaggregation significantly. While MDAV* and MDAV^*_γ incrementally improve the traditional class of maximum distance heuristics and deliver lower information loss in quadratic time, ONA* combines MDAV* with a new Lloyd-based post-processing and is able to lower information loss even further, without increasing the overall asymptotic time complexity. According to the experimental evaluation made as part of this thesis, ONA* and its variant ONA^*_γ are currently the most utility-preserving microaggregation heuristics suitable for all values of k. Their only contender for this title would be PCL, which is another Lloyd-based heuristic briefly discussed in section 6.2. However, as PCL is not applicable for values of $k \leq 100$ without further compromises in utility and time consumption, its practical relevance for microaggregation is limited. Besides the efforts to lower information loss within the same time frame, this thesis presents significant improvements for sub-quadratic time microaggregation heuristics. For situations in which a quadratic time complexity is no option, like for databases of several million elements, the algorithms MONA and MONA_2D presented in chapter 7 offer valid trade-offs between time consumption and information loss. They use adapted MONDRIAN variants as a pre-processing step and ONA* as the low-loss clustering algorithm that is applied, once database sizes have been reduced to manageable sizes.

The second main part of this thesis are new results regarding the complexity of microaggregation. By using a graph theoretic approach and a reduction from 3-SAT, NP-hardness of the k-anonymous microaggregation problem could be shown for all fixed values of $d \geq 2$ and $k \geq 4$. This is an important step assuring the necessity of heuristics, as previously NP-hardness was only shown for the case $k = 3$. In addition to these two main results, significant effort has been made to describe the model of microaggregation in a precise mathematical way, compare microaggregation to alternative privacy and utility definitions and to state and prove some key differences between the complexity and solution structure of the k-clustering and k-member clustering problems.

There are, however, still a lot of open questions. A mainly academic issue is the question whether k-anonymous microaggregation is NP-hard for fixed $d \geq 2$ and $k = 2$. While the answer to this question is of little relevance for the practical use of k-member clusterings as a way to anonymize microdata, from a purely mathematical perspective it is interesting

because there seem to be good reasons for both possible answers to this question.

On a more practical note, there is of course still room for advancement of microaggregation heuristics. Although the algorithms MDAV*, ONA* and MONA_2D, developed as part of this thesis, represent the state-of-the-art in microaggregation, there still seems to be a chance to lower information loss even further. This could be achieved by tuning parameters, introducing new splitting or re-clustering rules or combining them with new techniques. Especially interesting in this regard could be the alternative clustering techniques discussed in chapter 8.

Besides these minor open questions and iterative improvements to be made, there are two main issues that define the relevance and applicability of microaggregation in the future: approximation guarantees and the advancement of differential privacy. In chapter 9 and 10 these problems, reasons for their difficulty and first results are discussed briefly. Due to the fact that most microaggregation strategies that are successful in practice, cannot guarantee to find good solutions, there is a big gap in our understanding of approximation algorithms for this problem. Further, in contrast to the k-means clustering problem, no lower bounds for approximability are known for microaggregation. Following the path of LP-relaxation as discussed in section 9.3 could at least narrow this gap. Most of the microaggregation heuristics discussed in this thesis can in some way be combined to obtain better initial solutions, cut computational cost or improve upon results to further reduce information loss. Therefore, there could also be hope to combine new approximation strategies with established heuristics to efficiently find new clusterings with an even better trade-off between time consumption and information loss than possible today.

The invention of differential privacy has influenced most modern data privacy results. However, despite its popularity, there still is a lack of viable differentially private alternatives to private microdata release as offered by microaggregation. As differential privacy clearly offers the possibility of better and also provable privacy guarantees, there is a pressing need to combine both methods. As the work on safe k-anonymization (see section 3.3 and 10.2) shows, there is hope to use established microaggregation results without losing the option for differential privacy. Possible approaches have been outlined in chapter 10. As discussed in section 10.3 MONDRIAN seems to be a suitable candidate to start this undertaking.

Bibliography

[1] Ahmadian, S., Norouzi-Fard, A., Svensson, O., and Ward, J. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In: *SIAM Journal on Computing* 49(4):FOCS17–97, 2019.

[2] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. NP-hardness of Euclidean sum-of-squares clustering. In: *Machine learning* 75(2):245–248, 2009.

[3] Anwar, N *Micro-aggregation-the small aggregates method*. Tech. rep. Internal report. Luxembourg: Eurostat, 1993.

[4] Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2007, pp. 1027–1035.

[5] Barbaro, M. and Zeller, T. A Face Is Exposed for AOL Searcher No. 4417749. In: *The New York Times*, 2006. URL: https://www.nytimes.com/2006/08/09/technology/09aol.html.

[6] Barth-Jones, D. The 're-identification' of Governor William Weld's medical information: a critical re-examination of health data identification risks and privacy protections. In: *Then and Now*, 2012.

[7] Bayardo, R. J. and Agrawal, R. Data privacy through optimal k-anonymization. In: *21st International conference on data engineering (ICDE'05)*. IEEE. 2005, pp. 217–228.

[8] Blum, A., Ligett, K., and Roth, A. A Learning Theory Approach to Non-Interactive Database Privacy. In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Victoria, British Columbia, Canada: Association for Computing Machinery, 2008, pp. 609–618. DOI: 10.1145/1374376.1374464.

[9] Blum, A., Ligett, K., and Roth, A. A Learning Theory Approach to Noninteractive Database Privacy. In: *Journal of the ACM (JACM)* 60(2):1–25, 2013.

[10] Boros, E. and Hammer, P. L. On clustering problems with connected optima in Euclidean spaces. In: *Discrete Mathematics* 75(1-3):81–88, 1989.

[11] Brickell, J. and Shmatikov, V. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2008, pp. 70–78.

[12] Byun, J.-W., Kamra, A., Bertino, E., and Li, N. Efficient k-anonymization using clustering techniques. In: *International Conference on Database Systems for Advanced Applications*. Springer. 2007, pp. 188–200.

[13] Chang, C.-C., Li, Y.-C., and Huang, W.-H. TFRP: An efficient microaggregation algorithm for statistical disclosure control. In: *Journal of Systems and Software* 80(11):1866–1878, 2007.

[14] Defays, D and Anwar, M. Masking microdata using micro-aggregation. In: *Journal of Official Statistics* 14(4):449, 1998.

[15] Defays, D and Nanopoulos, P. Panels of enterprises and confidentiality: the small aggregates method. In: *Proceedings of the 1992 symposium on design and analysis of longitudinal surveys*. 1993, pp. 195–204.

[16] Domingo-Ferrer, J., Martínez-Ballesté, A., Mateo-Sanz, J. M., and Sebé, F. Efficient multivariate data-oriented microaggregation. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 15(4):355–369, 2006.

[17] Domingo-Ferrer, J. and Mateo-Sanz, J. M. *Reference data sets to test and compare SDC methods for protection of numerical microdata*. https://web.archive.org/web/20190412063606/http://neon.vb.cbs.nl/casc/CASCtestsets.htm. 2002.

[18] Domingo-Ferrer, J. and Mateo-Sanz, J. M. Practical data-oriented microaggregation for statistical disclosure control. In: *IEEE Transactions on Knowledge and Data Engineering* 14(1):189–201, 2002.

[19] Domingo-Ferrer, J., Sebé, F., and Solanas, A. A polynomial-time approximation to optimal multivariate microaggregation. In: *Computers & Mathematics with Applications* 55(4):714–732, 2008.

[20] Domingo-Ferrer, J. and Torra, V. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. In: *Data Mining and Knowledge Discovery* 11(2):195–212, 2005.

[21] Du, W., Foot, C., Moniot, M., Bray, A., and Groce, A. Differentially private confidence intervals. In: *arXiv preprint arXiv:2001.02285*, 2020.

[22] Dwork, C. Differential privacy: A survey of results. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[23] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[24] Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. In: *Foundations and Trends in Theoretical Computer Science* 9(3–4):211–407, 2014.

[25] European Association for Theoretical Computer Science *Gödel Prize 2017*. https://eatcs.org/index.php/goedel-prize. 2017.

[26] European Parliament and Council of the European Union *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[27] Feldman, D., Monemizadeh, M., and Sohler, C. A PTAS for k-means clustering based on weak coresets. In: *Proceedings of the twenty-third annual symposium on Computational geometry*. 2007, pp. 11–18.

[28] Fung, B. C., Wang, K., Chen, R., and Yu, P. S. Privacy-preserving data publishing: A survey of recent developments. In: *ACM Computing Surveys (CSUR)* 42(4):1–53, 2010.

[29] Garey, M. R. and Johnson, D. S. *Computers and Intractability*. Vol. 174. W.H. Freeman and Company, 1979.

[30] Golle, P. Revisiting the uniqueness of simple demographics in the US population. In: *Proceedings of the 5th ACM workshop on Privacy in electronic society*. ACM. 2006, pp. 77–80.

[31] Hansen, S. L. and Mukherjee, S. A polynomial algorithm for optimal univariate microaggregation. In: *IEEE Transactions on Knowledge and Data Engineering* 15(4):1043–1044, 2003.

[32] Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. 2004, pp. 291–300.

[33] Her Majesty, by and with the advice and consent of the Senate and House of Commons of Canada *Personal Information Protection and Electronic Documents Act (S.C. 2000, c. 5)*. 2000. URL: https://laws-lois.justice.gc.ca/eng/acts/P-8.6/.

[34] Inaba, M., Katoh, N., and Imai, H. Applications of Weighted Voronoi Diagrams and Randomization to Variance-based k-clustering: (Extended Abstract). In: *Proceedings of the Tenth Annual Symposium on Computational Geometry*. SCG '94. Stony Brook, New York, USA: ACM, 1994, pp. 332–339. DOI: 10.1145/177424.178042.

[35] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. In: *Computational Geometry* 28(2-3):89–112, 2004.

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

[37] LeFevre, K., DeWitt, D. J., and Ramakrishnan, R. Mondrian multidimensional k-anonymity. In: *22nd International conference on data engineering (ICDE'06)*. IEEE. 2006, pp. 25–25.

[38] Li, N., Li, T., and Venkatasubramanian, S. Closeness: A new privacy measure for data publishing. In: *IEEE Transactions on Knowledge and Data Engineering* 22(7):943–956, 2010.

[39] Li, N., Li, T., and Venkatasubramanian, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In: *2007 IEEE 23rd International Conference on Data Engineering*. IEEE. 2007, pp. 106–115.

[40] Li, N., Qardaji, W., and Su, D. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In: *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM. 2012, pp. 32–33.

[41] Li, T. and Li, N. On the tradeoff between privacy and utility in data publishing. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2009, pp. 517–526.

[42] Lichman, M. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml. 2013.

[43] Lichtenstein, D. Planar formulae and their uses. In: *SIAM Journal on Computing* 11(2):329–343, 1982.

[44] Lin, J.-L., Wen, T.-H., Hsieh, J.-C., and Chang, P.-C. Density-based microaggregation for statistical disclosure control. In: *Expert Systems with Applications* 37(4):3256–3263, 2010.

[45] Lloyd, S. P. Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28(2):129–137, 1982.

[46] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1(1):3, 2007.

[47] MacQueen, J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 14. University of California Press, 1967, pp. 281–297.

[48] Mahajan, M., Nimbhorkar, P., and Varadarajan, K. The planar k-means problem is NP-hard. In: *International Workshop on Algorithms and Computation*. Springer. 2009, pp. 274–285.

[49] Mahmood, A. N., Kabir, M. E., and Mustafa, A. K. New multi-dimensional sorting based k-anonymity microaggregation for statistical disclosure control. In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2012, pp. 256–272.

[50] Martínez, S., Sánchez, D., and Valls, A. Semantic adaptive microaggregation of categorical microdata. In: *Computers & Security* 31(5):653–672, 2012.

[51] Mateo Sanz, J. M. and Domingo Ferrer, J. A comparative study of microaggregation methods. In: *Qüestiió* 22(3), 1998.

[52] Matoušek, J. On approximate geometric k-clustering. In: *Discrete & Computational Geometry* 24(1):61–84, 2000.

[53] McSherry, F. and Talwar, K. Mechanism design via differential privacy. In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE. 2007, pp. 94–103.

[54] Meyerson, A. and Williams, R. On the complexity of optimal k-anonymity. In: *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2004, pp. 223–228.

[55] Micali, S. and Vazirani, V. V. An $\mathcal{O}(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In: *21st Annual Symposium on Foundations of Computer Science (SFCS 1980)*. IEEE. 1980, pp. 17–27.

[56] Narayanan, A. and Shmatikov, V. Robust de-anonymization of large sparse datasets. In: *2008 IEEE Symposium on Security and Privacy (SP 2008)*. IEEE. 2008, pp. 111–125.

[57] Oganian, A. and Domingo-Ferrer, J. On the complexity of optimal microaggregation for statistical disclosure control. In: *Statistical Journal of the United Nations Economic Commission for Europe* 18(4):345–353, 2001.

[58] Rebollo-Monedero, D., Forné, J., Pallarès, E., and Parra-Arnau, J. A modification of the Lloyd algorithm for k-anonymous quantization. In: *Information Sciences* 222:185–202, 2013.

[59] Samarati, P. and Sweeney, L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report. 1998.

[60] Sankar, L., Rajagopalan, S. R., and Poor, H. V. Utility-privacy tradeoffs in databases: An information-theoretic approach. In: *IEEE Transactions on Information Forensics and Security* 8(6):838–852, 2013.

[61] Solanas, A., Martinez-Balleste, A., and Domingo-Ferrer, J V-MDAV: a multivariate microaggregation with variable group size. In: *17th COMPSTAT Symposium of the IASC, Rome*. 2006, pp. 917–925.

[62] Soria-Comas, J. and Domingo-Ferrer, J. Differential privacy via t-closeness in data publishing. In: *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*. IEEE. 2013, pp. 27–35.

[63] Soria-Comas, J., Domingo-Ferrer, J., and Mulero, R. Efficient Near-Optimal Variable-Size Microaggregation. In: *International Conference on Modeling Decisions for Artificial Intelligence*. Springer. 2019, pp. 333–345.

[64] Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., and Martínez, S. Enhancing data utility in differential privacy via microaggregation-based k-anonymity. In: *The VLDB Journal—The International Journal on Very Large Data Bases* 23(5):771–794, 2014.

[65] Soria-Comas, J., Domingo-Ferrer, J., Sanchez, D., and Martinez, S. t-closeness through microaggregation: Strict privacy with enhanced utility preservation. In: *IEEE Transactions on Knowledge and Data Engineering* 27(11):3098–3110, 2015.

[66] Sweeney, L. k-anonymity: A model for protecting privacy. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10(05):557–570, 2002.

[67] Sweeney, L. Uniqueness of simple demographics in the US population. In: *LIDAP-WP4, 2000*, 2000.

[68] Thaeter, F. Hardness of k-anonymous microaggregation. In: *17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. Ed. by J. Hurink, S. Klootwijk, B. Manthey, V. Reijnders, and M. S. Uiterkamp. Enschede, 2019, pp. 135–138.

[69] Thaeter, F. and Reischuk, R. Hardness of k-anonymous microaggregation. In: *Discrete Applied Mathematics*, 2020. DOI: 10.1016/j.dam.2020.10.005.

[70] Thaeter, F. and Reischuk, R. Improving Anonymization Clustering. In: *SICHERHEIT 2018*. Ed. by H. Langweg, M. Meier, B. C. Witt, and D. Reinhardt. Bonn: Gesellschaft für Informatik e.V., 2018, pp. 69–82.

[71] Thaeter, F. and Reischuk, R. Scalable k-anonymous Microaggregation: Exploiting the Tradeoff between Computational Complexity and Information Loss. In: *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*. 2021, pp. 87–98.

[72] Torra, V. Microaggregation for categorical variables: a median based approach. In: *International Workshop on Privacy in Statistical Databases*. Springer. 2004, pp. 162–174.

[73] Vadhan, S. The complexity of differential privacy. In: *Tutorials on the Foundations of Cryptography*. Springer, 2017, pp. 347–450.

[74] Willenborg, L. and De Waal, T. *Elements of Statistical Disclosure Control*. Vol. 155. Springer Science & Business Media, 2012.

[75] Xu, J., Wang, W., Pei, J., Wang, X., Shi, B., and Fu, A. W.-C. Utility-based anonymization using local recoding. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2006, pp. 785–790.

A

Raw results of MDAV-like heuristics

Table A.1: Information losses of maximum distance heuristics on small databases ($n \leq 4092$) from the CASC project. Census $\in \mathcal{X}_{1080,13}$, Tarragona $\in \mathcal{X}_{834,13}$, EIA $\in \mathcal{X}_{4092,11}$.

	Information Loss in % on Census					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	3.18	5.72	7.64	9.00	11.54	14.40
MDAV	3.18	5.69	7.49	9.09	11.60	14.16
MDAV$^+$	3.16	5.66	7.51	9.01	11.66	14.07
V-MDAV	3.16	5.66	7.51	8.98	11.59	14.04
MDAV*	3.16	5.78	7.45	8.83	11.37	14.00
MDAV$^*_\gamma$	3.11	5.59	7.24	8.61	11.04	13.77

	Information Loss in % on Tarragona					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	9.31	16.98	19.54	22.53	27.52	33.18
MDAV	9.33	16.93	19.55	22.46	27.52	33.19
MDAV$^+$	9.29	16.95	19.77	22.87	28.26	33.25
V-MDAV	9.29	15.85	19.70	22.87	28.25	33.25
MDAV*	9.44	16.15	19.19	22.26	28.40	34.74
MDAV$^*_\gamma$	9.28	16.09	19.19	21.96	27.76	32.97

	Information Loss in % on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	0.27	0.47	0.77	1.67	2.28	3.71
MDAV	0.31	0.48	0.67	1.67	2.17	3.84
MDAV$^+$	0.32	0.49	0.67	1.78	2.21	3.55
V-MDAV	0.23	0.46	0.67	1.06	2.21	2.79
MDAV*	0.22	0.45	0.62	0.91	2.03	2.63
MDAV$^*_\gamma$	0.20	0.39	0.54	0.82	1.66	2.18

Table A.2: Information losses of maximum distance heuristics on small databases ($n = 1024$) from the UCI repository. Cloud1 $\in \mathcal{X}_{1024,10}$, Cloud2 $\in \mathcal{X}_{1024,10}$.

	Information Loss in % on Cloud1					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	1.21	2.18	3.42	4.23	5.59	7.35
MDAV	1.21	2.22	3.74	4.31	5.70	7.05
MDAV$^+$	1.20	2.23	3.78	4.31	5.56	6.98
V-MDAV	1.19	2.15	3.78	4.29	5.53	6.94
MDAV*	1.16	2.10	3.65	4.09	5.54	6.70
MDAV$^*_\gamma$	1.15	2.10	3.63	4.08	5.49	6.70

	Information Loss in % on Cloud2					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	0.68	1.19	1.63	2.04	2.66	3.36
MDAV	0.68	1.21	1.70	2.03	2.69	3.40
MDAV$^+$	0.66	1.22	1.62	2.09	2.78	3.50
V-MDAV	0.66	1.19	1.62	2.05	2.77	3.46
MDAV*	0.64	1.09	1.52	1.87	2.50	3.28
MDAV$^*_\gamma$	0.63	1.09	1.48	1.86	2.48	3.23

Table A.3: Information losses of maximum distance heuristics on large databases ($n \geq 30000$) from the UCI repository. MD is not able to handle the Adult database, as it is too big to be handled with quadratic space requirements. Credit Card $\in \mathcal{X}_{30000,24}$, Adult $\in \mathcal{X}_{48842,3}$.

	Information Loss in % on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	3.66	6.43	8.56	10.23	12.64	15.16
MDAV	3.66	6.40	8.53	10.17	12.58	15.18
MDAV$^+$	3.65	6.39	8.49	10.17	12.59	15.21
V-MDAV	3.64	6.38	8.49	10.17	12.58	15.17
MDAV*	3.65	6.44	8.48	10.22	12.36	14.67
MDAV$^*_\gamma$	3.59	6.25	8.25	9.83	12.11	14.47

	Information Loss in % on Adult					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	–	–	–	–	–	–
MDAV	0.054	0.108	0.162	0.217	0.317	0.463
MDAV$^+$	0.053	0.107	0.163	0.217	0.317	0.464
V-MDAV	0.046	0.102	0.158	0.210	0.314	0.464
MDAV*	0.041	0.088	0.138	0.184	0.281	0.421
MDAV$^*_\gamma$	0.040	0.086	0.133	0.180	0.273	0.411

Table A.4: Mean information losses of maximum distance heuristics on sets of synthetic databases. SimU $= \{\text{SimU}_0, \ldots, \text{SimU}_{24}\}, \text{SimU}_i \in \mathcal{X}_{1000,10}$, SimC $= \{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$.

	Mean Information Loss in % on SimU					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	10.16	18.19	23.63	27.99	34.31	40.95
MDAV	10.12	18.15	23.65	28.07	34.58	41.03
MDAV$^+$	10.14	18.03	23.62	27.90	34.04	40.63
V-MDAV	10.13	17.99	23.55	27.80	34.01	40.43
MDAV*	10.12	17.76	23.10	27.28	33.37	39.66
MDAV$^*_\gamma$	10.05	17.63	23.03	27.12	33.31	39.56
	Mean Information Loss in % on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	3.88	7.00	9.32	11.33	14.51	18.37
MDAV	3.86	6.93	9.32	11.26	14.46	18.45
MDAV$^+$	3.82	6.86	9.23	11.04	14.11	18.09
V-MDAV	3.32	6.19	8.35	10.16	13.14	16.97
MDAV*	3.36	6.00	8.17	9.66	12.32	15.69
MDAV$^*_\gamma$	3.28	5.73	7.61	9.16	11.65	14.84

Table A.5: Runtimes in s of maximum distance heuristics on small databases ($n \leq 4092$) from the CASC project. Census $\in \mathcal{X}_{1080,13}$, Tarragona $\in \mathcal{X}_{834,13}$, EIA $\in \mathcal{X}_{4092,11}$.

	Runtime in s on Census					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	0.09	0.03	0.02	0.02	0.01	0.01
MDAV	0.04	0.01	0.01	0.01	0.01	0.01
MDAV$^+$	0.03	0.01	0.01	0.01	0.01	0.00
V-MDAV	0.04	0.02	0.01	0.01	0.01	0.01
MDAV*	0.08	0.04	0.03	0.03	0.02	0.02
MDAV$^*_\gamma$	0.03	0.02	0.02	0.01	0.01	0.01
	Runtime in s on Tarragona					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	0.07	0.02	0.01	0.01	0.01	0.01
MDAV	0.03	0.01	0.01	0.01	0.01	0.00
MDAV$^+$	0.03	0.01	0.01	0.01	0.01	0.00
V-MDAV	0.03	0.01	0.01	0.01	0.01	0.00
MDAV*	0.06	0.03	0.02	0.02	0.02	0.02
MDAV$^*_\gamma$	0.02	0.01	0.01	0.01	0.01	0.01
	Runtime in s on EIA					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MD	2.38	1.58	1.15	0.91	0.67	0.49
MDAV	0.23	0.15	0.10	0.09	0.07	0.06
MDAV$^+$	0.11	0.06	0.05	0.04	0.03	0.02
V-MDAV	0.15	0.13	0.10	0.11	0.08	0.13
MDAV*	0.37	0.24	0.18	0.14	0.10	0.09
MDAV$^*_\gamma$	0.30	0.22	0.18	0.18	0.18	0.14

Table A.6: Runtimes in s of maximum distance heuristics on large databases ($n \geq 30000$) from the UCI repository. MD is not able to handle the Adult database, as it is too big to be handled with quadratic space requirements. Credit Card $\in \mathcal{X}_{30000,24}$, Adult $\in \mathcal{X}_{48842,3}$.

	Runtime in s on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	895.1	590.3	441.1	354.6	256.0	182.9
MDAV	16.4	11.8	9.4	8.6	6.2	5.3
MDAV$^+$	6.7	4.6	3.3	2.7	2.0	1.5
V-MDAV	15.4	11.4	9.9	8.9	8.1	7.3
MDAV*	24.5	17.2	14.4	12.7	10.8	9.2
MDAV$^*_\gamma$	28.2	19.3	16.0	13.7	12.1	9.8

	Runtime in s on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	–	–	–	–	–	–
MDAV	12.3	7.8	5.9	4.8	3.3	2.4
MDAV$^+$	9.4	6.0	4.6	3.9	2.8	2.2
V-MDAV	18.7	15.4	14.1	12.6	11.0	9.9
MDAV*	28.6	20.7	15.3	13.1	9.8	7.9
MDAV$^*_\gamma$	33.0	24.3	18.8	16.7	12.9	11.7

Table A.7: Mean runtimes in s of maximum distance heuristics on sets of synthetic databases. SimU $= \{\text{SimU}_0, \ldots, \text{SimU}_{24}\}, \text{SimU}_i \in \mathcal{X}_{1000,10}$, SimC $= \{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$.

	Mean Runtime in s on SimU					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	0.03	0.02	0.02	0.01	0.01	0.01
MDAV	0.01	0.01	0.00	0.00	0.00	0.00
MDAV$^+$	0.01	0.00	0.00	0.00	0.00	0.00
V-MDAV	0.01	0.01	0.01	0.01	0.01	0.01
MDAV*	0.02	0.02	0.01	0.01	0.01	0.01
MDAV$^*_\gamma$	0.02	0.01	0.01	0.01	0.01	0.01

	Mean Runtime in s on SimC					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MD	0.12	0.08	0.06	0.05	0.04	0.03
MDAV	0.02	0.02	0.01	0.01	0.01	0.01
MDAV$^+$	0.01	0.01	0.01	0.01	0.00	0.00
V-MDAV	0.02	0.02	0.02	0.02	0.02	0.02
MDAV*	0.05	0.03	0.03	0.02	0.02	0.02
MDAV$^*_\gamma$	0.05	0.03	0.03	0.03	0.03	0.02

Table A.8: Average cluster sizes for variable-size maximum distance heuristics.

	Average cluster size on Census					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.00	3.00	4.00	5.02	7.01	10.09
MDAV*	2.19	3.42	4.50	5.71	8.06	11.49
MDAV$^*_\gamma$	2.06	3.04	4.41	5.45	7.61	10.29

	Average cluster size on Tarragona					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.00	3.02	4.03	5.02	7.25	10.17
MDAV*	2.15	3.39	4.53	5.67	8.26	11.42
MDAV$^*_\gamma$	2.04	3.22	4.56	5.25	7.13	10.56

	Average cluster size on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.17	3.61	4.00	5.54	7.01	11.46
MDAV*	2.24	3.38	4.24	5.62	7.56	10.88
MDAV$^*_\gamma$	2.19	3.37	4.60	6.14	9.39	12.00

	Average cluster size on Cloud1					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.00	3.14	4.16	5.02	7.01	10.04
MDAV*	2.17	3.32	4.47	5.48	7.70	10.89
MDAV$^*_\gamma$	2.13	3.32	4.32	5.36	7.31	10.89

	Average cluster size on Cloud2					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.00	3.04	4.00	5.02	7.06	10.56
MDAV*	2.24	3.34	4.47	5.60	7.64	10.78
MDAV$^*_\gamma$	2.11	3.21	4.30	5.69	7.47	10.67

	Average cluster size on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.02	3.02	4.00	5.00	7.03	10.04
MDAV*	2.21	3.50	4.75	6.01	8.45	11.94
MDAV$^*_\gamma$	2.11	3.23	4.35	5.46	7.99	11.30

	Average cluster size on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	2.10	3.10	4.12	5.10	7.10	10.11
MDAV*	2.12	3.36	4.23	5.39	7.41	10.72
MDAV$^*_\gamma$	2.09	3.38	4.35	5.60	7.70	11.45

Table A.9: Gain factors γ used for V-MDAV and MDAV^*_γ to achieve information losses reported in table A.1 to table A.4. For SimU and SimC gain factors are chosen for individual databases and mean γ is reported here.

	γ used on Census					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.0	0.0	0.0	0.2	0.1	0.2
MDAV^*_γ	0.8	0.6	0.9	0.9	0.9	0.8
	γ used on Tarragona					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.0	0.3	0.3	0.0	0.6	0.3
MDAV^*_γ	0.8	0.9	1.0	0.8	0.7	0.8
	γ used on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.2	0.6	0.0	0.4	0.0	1.3
MDAV^*_γ	0.8	0.7	0.8	1.2	1.1	1.2
	γ used on Cloud1					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.1	0.4	0.4	0.2	0.1	0.2
MDAV^*_γ	0.9	1.0	0.9	0.9	0.8	1.0
	γ used on Cloud2					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.0	0.2	0.0	0.1	0.2	0.5
MDAV^*_γ	0.8	0.8	0.8	1.0	0.9	0.9
	γ used on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.1	0.1	0.0	0.0	0.1	0.1
MDAV^*_γ	0.8	0.8	0.8	0.8	0.9	0.9
	γ used on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.4	0.1	0.4	0.2	0.2	0.2
MDAV^*_γ	0.7	0.9	0.9	1.0	0.9	1.0
	Mean γ used on SimU					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.1	0.2	0.2	0.2	0.1	0.2
MDAV^*_γ	0.9	0.9	0.9	0.9	1.0	1.0
	Mean γ used on SimC					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
V-MDAV	0.2	0.4	0.5	0.5	0.7	0.8
MDAV^*_γ	0.8	0.8	0.8	0.8	0.9	1.1

B

Raw results of Lloyd-based heuristics

Table B.1: Information losses of Lloyd-based heuristics on small databases ($n \leq 4092$) from the CASC project. Census $\in \mathcal{X}_{1080,13}$, Tarragona $\in \mathcal{X}_{834,13}$, EIA $\in \mathcal{X}_{4092,11}$. Please take results for $\texttt{LMaS}_{10}$ and $\texttt{ONA}_{10}$ with a grain of salt, as due to the low number of repetitions these results might vary.

	Information Loss in % on Census					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	3.47	6.00	7.71	9.14	11.39	14.27
$\texttt{LMaS}_{100}$	3.37	5.78	7.49	8.84	11.00	13.81
$\texttt{LMaS}_{1000}$	3.37	5.76	7.47	8.86	10.89	13.50
$\texttt{ONA}_{10}$	3.51	5.61	7.02	8.25	10.28	12.32
$\texttt{ONA}_{100}$	3.47	5.46	7.01	8.18	10.15	12.35
$\texttt{ONA}_{1000}$	3.41	5.43	6.89	8.10	9.95	12.27
$\texttt{ONA}^{*}$	3.06	5.27	6.71	8.04	10.07	12.46
$\texttt{ONA}^{*}_{\gamma}$	3.06	5.22	6.72	7.91	9.84	12.31
	Information Loss in % on Tarragona					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	9.56	16.84	18.63	22.81	28.15	33.90
$\texttt{LMaS}_{100}$	9.46	15.83	18.73	21.83	27.61	31.83
$\texttt{LMaS}_{1000}$	9.20	15.79	18.33	21.67	27.25	31.78
$\texttt{ONA}_{10}$	9.72	15.43	17.72	21.11	26.85	31.43
$\texttt{ONA}_{100}$	9.49	15.01	17.83	20.84	26.35	30.87
$\texttt{ONA}_{1000}$	9.17	14.86	17.56	20.78	26.42	30.84
$\texttt{ONA}^{*}$	9.06	15.11	17.79	20.48	26.34	31.15
$\texttt{ONA}^{*}_{\gamma}$	9.05	15.10	17.76	20.49	26.12	30.51
	Information Loss in % on EIA					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	0.19	0.40	0.57	0.81	1.73	2.09
$\texttt{LMaS}_{100}$	0.19	0.39	0.56	0.80	1.67	2.04
$\texttt{LMaS}_{1000}$	0.19	0.39	0.55	0.79	1.66	2.03
$\texttt{ONA}_{10}$	0.22	0.41	0.60	0.84	1.66	2.03
$\texttt{ONA}_{100}$	0.21	0.40	0.59	0.80	1.60	2.01
$\texttt{ONA}_{1000}$	0.21	0.40	0.59	0.79	1.59	1.99
$\texttt{ONA}^{*}$	0.20	0.37	0.52	0.79	1.63	1.99
$\texttt{ONA}^{*}_{\gamma}$	0.19	0.37	0.52	0.78	1.58	1.98

Table B.2: Information losses of Lloyd-based heuristics on small databases ($n = 1024$) from the UCI repository. Cloud1 $\in \mathcal{X}_{1024,10}$, Cloud2 $\in \mathcal{X}_{1024,10}$. Please take results for $\texttt{LMaS}_{10}$ and $\texttt{ONA}_{10}$ with a grain of salt, as due to the low number of repetitions these results might vary.

	Information Loss in % on Cloud1					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	1.25	2.23	3.82	4.39	5.53	6.92
$\texttt{LMaS}_{100}$	1.23	2.19	3.46	4.27	5.51	6.82
$\texttt{LMaS}_{1000}$	1.23	2.20	3.39	4.25	5.43	6.78
$\texttt{ONA}_{10}$	1.23	2.15	3.20	3.89	5.19	6.41
$\texttt{ONA}_{100}$	1.20	2.13	3.20	3.82	4.95	6.37
$\texttt{ONA}_{1000}$	1.20	2.11	3.15	3.81	4.97	6.35
$\texttt{ONA}^{*}$	1.15	2.02	3.24	3.92	5.07	6.28
$\texttt{ONA}^{*}_{\gamma}$	1.13	2.02	3.10	3.68	4.92	6.28

	Information Loss in % on Cloud2					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	0.66	1.15	1.56	1.89	2.45	3.17
$\texttt{LMaS}_{100}$	0.66	1.13	1.54	1.87	2.43	3.14
$\texttt{LMaS}_{1000}$	0.65	1.13	1.51	1.84	2.40	3.11
$\texttt{ONA}_{10}$	0.68	1.11	1.46	1.78	2.28	3.00
$\texttt{ONA}_{100}$	0.67	1.10	1.46	1.74	2.27	2.96
$\texttt{ONA}_{1000}$	0.67	1.10	1.44	1.72	2.23	2.93
$\texttt{ONA}^{*}$	0.61	1.04	1.40	1.71	2.22	2.92
$\texttt{ONA}^{*}_{\gamma}$	0.60	1.03	1.39	1.67	2.19	2.91

Table B.3: Information losses of Lloyd-based heuristics on large databases ($n \geq 30000$) from the UCI repository. Credit Card $\in \mathcal{X}_{30000,24}$, Adult $\in \mathcal{X}_{48842,3}$. Please take results for $\texttt{LMaS}_{10}$ and $\texttt{ONA}_{10}$ with a grain of salt, as due to the low number of repetitions these results might vary. $\texttt{LMaS}_{1000}$ as well as $\texttt{ONA}_{100}$ and $\texttt{ONA}_{1000}$ have not been tested as they require too much time.

	Information Loss in % on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	3.89	6.53	7.95	9.32	11.50	13.48
$\texttt{LMaS}_{100}$	3.89	6.48	7.92	9.28	11.36	13.42
$\texttt{ONA}_{10}$	3.98	6.16	7.58	8.72	10.53	12.30
$\texttt{ONA}^{*}$	3.50	5.86	7.53	8.65	10.23	12.24
$\texttt{ONA}^{*}_{\gamma}$	3.50	5.86	7.40	8.46	10.20	12.12

	Information Loss in % on Adult					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	0.042	0.090	0.137	0.186	0.281	0.418
$\texttt{LMaS}_{100}$	0.041	0.090	0.137	0.183	0.275	0.408
$\texttt{ONA}_{10}$	0.047	0.094	0.139	0.182	0.262	0.374
$\texttt{ONA}^{*}$	0.039	0.081	0.123	0.165	0.243	0.357
$\texttt{ONA}^{*}_{\gamma}$	0.039	0.081	0.122	0.161	0.241	0.353

Table B.4: Mean information losses of Lloyd-based heuristics on sets of synthetic databases. SimU $= \{\text{SimU}_0, \ldots, \text{SimU}_{24}\}, \text{SimU}_i \in \mathcal{X}_{1000,10}$, SimC $= \{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$. Please take results for $\texttt{LMaS}_{10}$ and $\texttt{ONA}_{10}$ with a grain of salt, as due to the low number of repetitions these results might vary. Means are taken over the results with lowest IL on each database.

	Mean Information Loss in % on SimU					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	11.96	19.21	24.08	27.65	32.85	38.31
$\texttt{LMaS}_{100}$	11.81	18.94	23.79	27.41	32.47	37.74
$\texttt{LMaS}_{1000}$	11.70	18.82	23.56	27.11	32.24	37.45
$\texttt{ONA}_{10}$	11.86	17.65	21.70	24.94	29.76	35.29
$\texttt{ONA}_{100}$	11.73	17.45	21.50	24.68	29.60	34.93
$\texttt{ONA}_{1000}$	11.60	17.31	21.36	24.49	29.36	34.72
$\texttt{ONA}^*$	10.02	16.69	20.95	24.26	29.05	34.41
$\texttt{ONA}^*_\gamma$	9.98	16.58	20.85	24.13	28.98	34.23

	Mean Information Loss in % on SimC					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	3.53	5.51	6.94	8.19	10.20	12.82
$\texttt{LMaS}_{100}$	3.45	5.44	6.87	8.08	10.06	12.57
$\texttt{LMaS}_{1000}$	3.43	5.38	6.81	8.01	9.94	12.40
$\texttt{ONA}_{10}$	3.62	5.48	6.77	7.81	9.46	11.69
$\texttt{ONA}_{100}$	3.54	5.40	6.69	7.72	9.36	11.54
$\texttt{ONA}_{1000}$	3.50	5.33	6.63	7.66	9.30	11.45
$\texttt{ONA}^*$	3.16	5.20	6.59	7.68	9.33	11.40
$\texttt{ONA}^*_\gamma$	3.14	5.15	6.51	7.55	9.20	11.24

Table B.5: Runtimes in s of Lloyd-based heuristics on small databases ($n \leq 4092$) from the CASC project. Census $\in \mathcal{X}_{1080,13}$, Tarragona $\in \mathcal{X}_{834,13}$, EIA $\in \mathcal{X}_{4092,11}$.

	Runtime in s on Census					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	0.28	0.19	0.16	0.14	0.13	0.11
$\texttt{LMaS}_{100}$	2.06	1.73	1.44	1.29	1.12	0.97
$\texttt{LMaS}_{1000}$	19.12	16.39	14.40	13.03	11.28	9.66
$\texttt{ONA}_{10}$	0.67	0.55	0.58	0.55	0.74	0.64
$\texttt{ONA}_{100}$	5.77	5.67	5.76	5.77	5.98	6.51
$\texttt{ONA}_{1000}$	55.97	60.44	61.10	59.82	64.64	66.62
$\texttt{ONA}^{*}$	0.13	0.06	0.05	0.04	0.03	0.04
$\texttt{ONA}^{*}_{\gamma}$	0.05	0.05	0.03	0.03	0.04	0.08

	Runtime in s on Tarragona					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	0.25	0.16	0.15	0.15	0.12	0.08
$\texttt{LMaS}_{100}$	1.77	1.60	1.37	1.22	1.06	0.78
$\texttt{LMaS}_{1000}$	16.84	14.53	13.10	11.96	9.67	7.61
$\texttt{ONA}_{10}$	0.54	0.45	0.53	0.53	0.66	0.72
$\texttt{ONA}_{100}$	4.44	4.92	5.18	5.68	6.90	8.29
$\texttt{ONA}_{1000}$	44.30	47.60	51.69	55.93	64.92	82.95
$\texttt{ONA}^{*}$	0.54	0.04	0.03	0.03	0.03	0.04
$\texttt{ONA}^{*}_{\gamma}$	0.04	0.03	0.03	0.02	0.04	0.05

	Runtime in s on EIA					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
$\texttt{LMaS}_{10}$	2.99	2.04	2.17	1.77	1.67	1.50
$\texttt{LMaS}_{100}$	24.86	21.95	19.06	18.24	15.14	11.94
$\texttt{LMaS}_{1000}$	243.84	213.34	189.83	176.39	150.34	117.41
$\texttt{ONA}_{10}$	7.77	10.44	6.16	14.66	2.46	2.13
$\texttt{ONA}_{100}$	78.41	97.59	56.13	147.08	24.84	21.68
$\texttt{ONA}_{1000}$	837.56	1,105.83	544.66	1,554.26	254.84	231.22
$\texttt{ONA}^{*}$	0.60	0.40	0.29	0.27	0.21	0.17
$\texttt{ONA}^{*}_{\gamma}$	0.51	0.38	0.30	0.28	0.32	0.21

Table B.6: Runtimes in s of Lloyd-based heuristics on large databases ($n \geq 30000$) from the UCI repository. Credit Card $\in \mathcal{X}_{30000,24}$, Adult $\in \mathcal{X}_{48842,3}$. $\texttt{LMaS}_{1000}$ as well as $\texttt{ONA}_{100}$ and $\texttt{ONA}_{1000}$ have not been tested as they require too much time.

	Runtime in s on Credit Card					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	599.0	560.2	481.8	418.5	339.0	255.9
$\texttt{LMaS}_{100}$	5438.9	4670.9	4138.9	3527.3	2,821.5	2,207.2
$\texttt{ONA}_{10}$	961.9	1333.4	1188.5	1041.8	958.8	869.7
$\texttt{ONA}^{*}$	49.3	42.9	35.4	36.5	32.8	29.6
$\texttt{ONA}^{*}_{\gamma}$	52.1	58.9	52.4	41.9	35.5	25.2

	Runtime in s on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	117.1	75.3	58.4	49.7	49.0	46.0
$\texttt{LMaS}_{100}$	995.7	685.3	529.8	455.3	471.4	413.9
$\texttt{ONA}_{10}$	5807.3	4859.9	4366.0	3203.9	2958.7	2482.2
$\texttt{ONA}^{*}$	226.3	148.6	116.4	99.8	78.6	52.3
$\texttt{ONA}^{*}_{\gamma}$	196.0	136.0	105.4	92.5	72.8	52.2

Table B.7: Mean runtimes in s of Lloyd-based heuristics on sets of synthetic databases. $\text{SimU} = \{\text{SimU}_0, \ldots, \text{SimU}_{24}\}, \text{SimU}_i \in \mathcal{X}_{1000,10}$, $\text{SimC} = \{\text{SimC}_0, \ldots, \text{SimC}_{24}\}$. Sizes of SimC_i are between $n = 1528$ and $n = 1765$ for $d = 10$. Means are taken over the results with lowest IL on each database.

	Mean Runtime in s on SimU					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	0.13	0.10	0.09	0.08	0.07	0.06
$\texttt{LMaS}_{100}$	1.08	0.90	0.82	0.74	0.65	0.59
$\texttt{LMaS}_{1000}$	10.80	9.36	8.19	7.50	6.50	5.89
$\texttt{ONA}_{10}$	0.53	0.55	0.56	0.54	0.55	0.55
$\texttt{ONA}_{100}$	4.78	5.06	5.07	5.04	5.02	4.92
$\texttt{ONA}_{1000}$	46.41	49.60	50.21	25.32	50.60	48.88
$\texttt{ONA}^{*}$	0.03	0.03	0.03	0.03	0.03	0.03
$\texttt{ONA}^{*}_{\gamma}$	0.03	0.03	0.03	0.03	0.03	0.06

	Mean Runtime in s on SimC					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
$\texttt{LMaS}_{10}$	0.28	0.18	0.13	0.12	0.10	0.09
$\texttt{LMaS}_{100}$	2.55	1.67	1.22	1.11	0.96	0.83
$\texttt{LMaS}_{1000}$	25.48	17.13	12.55	11.69	10.18	8.61
$\texttt{ONA}_{10}$	1.38	1.27	1.14	1.02	0.87	0.70
$\texttt{ONA}_{100}$	13.31	11.72	10.15	9.27	7.74	6.23
$\texttt{ONA}_{1000}$	130.25	117.87	104.38	93.46	77.78	64.23
$\texttt{ONA}^{*}$	0.09	0.07	0.06	0.05	0.05	0.05
$\texttt{ONA}^{*}_{\gamma}$	0.08	0.07	0.07	0.06	0.06	0.05

Table B.8: Gain factors γ used for $\texttt{ONA}^*_\gamma$ to achieve information losses reported in table B.1 to table B.4. For SimU and SimC gain factors are chosen for individual databases and mean γ is reported here.

	γ used for $\texttt{ONA}^*_\gamma$					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
Census	0.9	1.2	1.0	1.1	1.2	1.7
Tarragona	0.7	1.9	1.1	1.0	1.1	1.3
EIA	1.6	1.0	0.9	0.5	1.8	1.3
Cloud1	1.3	1.0	1.5	1.1	1.1	1.0
Cloud2	1.1	0.9	1.0	0.8	1.4	1.9
Credit Card	1.1	1.0	1.7	1.2	1.1	0.9
Adult	1.0	0.9	0.9	1.2	1.1	1.1
SimU	0.9	1.1	1.1	1.0	1.0	1.0
SimC	1.1	1.2	1.0	1.0	0.9	0.7

C

Raw results of Mondrian-based heuristics

Table C.1: Information losses of Mondrian-based heuristics on the Adult database. Adult $\in \mathcal{X}_{48842,3}$

	Information Loss in % on Adult					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	0.249	0.508	0.508	0.508	0.917	0.917
MONDRIAN_V	0.206	0.407	0.407	0.407	0.757	0.757
MONDRIAN_V2D	0.188	0.387	0.387	0.387	0.705	0.705
$\text{MONA}_{0.1}$	0.206	0.407	0.407	0.407	0.757	0.757
$\text{MONA\_2D}_{0.1}$	0.188	0.387	0.387	0.387	0.705	0.705
$\text{MONA}_{0.2}$	0.105	0.407	0.407	0.407	0.757	0.757
$\text{MONA\_2D}_{0.2}$	0.099	0.387	0.387	0.387	0.705	0.705
$\text{MONA}_{0.3}$	0.067	0.140	0.211	0.285	0.434	0.649
$\text{MONA\_2D}_{0.3}$	0.068	0.142	0.216	0.290	0.425	0.646
$\text{MONA}_{0.4}$	0.060	0.124	0.187	0.248	0.381	0.569
$\text{MONA\_2D}_{0.4}$	0.060	0.125	0.190	0.250	0.376	0.561
$\text{MONA}_{0.5}$	0.050	0.106	0.161	0.211	0.322	0.465
$\text{MONA\_2D}_{0.5}$	0.050	0.102	0.156	0.206	0.302	0.456
$\text{MONA}_{0.6}$	0.048	0.099	0.149	0.198	0.303	0.439
$\text{MONA\_2D}_{0.6}$	0.048	0.097	0.149	0.195	0.286	0.430
$\text{MONA}_{0.7}$	0.043	0.091	0.137	0.182	0.276	0.402
$\text{MONA\_2D}_{0.7}$	0.042	0.090	0.137	0.182	0.268	0.399
$\text{MONA}_{0.8}$	0.041	0.089	0.132	0.175	0.264	0.386
$\text{MONA\_2D}_{0.8}$	0.042	0.087	0.132	0.176	0.257	0.381
$\text{MONA}_{0.9}$	0.040	0.085	0.129	0.171	0.258	0.372
$\text{MONA\_2D}_{0.9}$	0.041	0.084	0.128	0.170	0.247	0.372
MONA_{1}	0.039	0.081	0.123	0.165	0.243	0.357

Table C.2: Information losses of Mondrian-based heuristics on the Credit Card database. Credit Card $\in \mathcal{X}_{30000,24}$

	Information Loss in % on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MONDRIAN	30.33	30.33	41.47	41.47	43.75	50.71
MONDRIAN_V	24.05	24.05	32.54	32.54	34.12	39.27
MONDRIAN_V2D	17.85	17.85	24.52	24.52	25.78	30.30
$\text{MONA}_{0.1}$	24.05	24.05	32.54	32.54	34.12	39.27
$\text{MONA\_2D}_{0.1}$	17.85	17.85	24.52	24.52	25.78	30.30
$\text{MONA}_{0.2}$	19.46	22.50	32.54	32.54	34.12	39.27
$\text{MONA\_2D}_{0.2}$	14.83	17.11	24.52	24.52	25.78	30.30
$\text{MONA}_{0.3}$	10.85	17.35	22.26	25.63	31.67	39.27
$\text{MONA\_2D}_{0.3}$	8.76	13.92	17.87	20.50	24.84	30.30
$\text{MONA}_{0.4}$	8.67	13.91	17.71	20.53	24.77	29.18
$\text{MONA\_2D}_{0.4}$	7.38	11.78	14.93	17.36	20.97	24.62
$\text{MONA}_{0.5}$	7.74	12.56	15.99	18.53	22.45	26.59
$\text{MONA\_2D}_{0.5}$	6.87	10.96	13.89	16.16	19.50	22.95
$\text{MONA}_{0.6}$	6.36	10.14	12.90	14.95	17.96	21.44
$\text{MONA\_2D}_{0.6}$	5.73	9.20	11.58	13.45	16.29	19.18
$\text{MONA}_{0.7}$	5.64	9.14	11.52	13.41	16.27	19.22
$\text{MONA\_2D}_{0.7}$	5.27	8.47	10.75	12.42	14.97	17.54
$\text{MONA}_{0.8}$	4.38	7.29	9.42	10.92	13.19	15.54
$\text{MONA\_2D}_{0.8}$	4.56	7.35	9.35	10.77	12.82	15.02
$\text{MONA}_{0.9}$	4.07	6.79	8.79	10.15	12.26	14.48
$\text{MONA\_2D}_{0.9}$	4.18	6.83	8.68	10.07	11.95	13.91
MONA_{1}	3.50	5.86	7.53	8.65	10.23	12.24

Table C.3: Information losses of Mondrian-based heuristics on the Winnipeg database. Winnipeg $\in \mathcal{X}_{325834,174}$

	Information Loss in % on Winnipeg					
	$k=2$	$k=3$	$k=4$	$k=5$	$k=7$	$k=10$
MONDRIAN	49.43	65.33	65.33	75.71	75.71	81.56
MONDRIAN_V	46.53	61.93	61.93	72.51	72.51	78.72
MONDRIAN_V2D	44.26	59.60	59.60	70.24	70.24	76.70
$\text{MONA}_{0.1}$	46.53	61.93	61.93	72.51	72.51	78.72
$\text{MONA\_2D}_{0.1}$	44.26	59.60	59.60	70.24	70.24	76.70
$\text{MONA}_{0.2}$	33.64	48.80	58.39	72.51	72.51	78.72
$\text{MONA\_2D}_{0.2}$	32.69	47.25	56.41	70.24	70.24	76.70
$\text{MONA}_{0.3}$	30.01	44.15	52.50	57.98	64.91	70.93
$\text{MONA\_2D}_{0.3}$	29.37	42.96	50.98	56.32	63.09	69.03
$\text{MONA}_{0.4}$	24.75	37.43	45.87	51.85	59.52	65.97
$\text{MONA\_2D}_{0.4}$	24.29	36.52	44.64	50.48	58.01	64.40
$\text{MONA}_{0.5}$	22.25	33.47	41.04	46.84	55.09	62.56
$\text{MONA\_2D}_{0.5}$	21.76	32.58	39.87	45.52	53.72	61.07
$\text{MONA}_{0.6}$	18.39	27.37	33.14	37.14	42.68	48.65
$\text{MONA\_2D}_{0.6}$	18.01	26.79	32.10	35.84	41.24	47.07
$\text{MONA}_{0.7}$	15.64	23.38	28.30	31.68	36.34	40.74
$\text{MONA\_2D}_{0.7}$	15.34	22.85	27.38	30.59	34.94	39.01
$\text{MONA}_{0.8}$	13.66	20.43	24.80	27.82	31.92	35.60
$\text{MONA\_2D}_{0.8}$	13.46	19.99	24.05	26.90	30.70	34.24

Table C.4: Runtimes in s of Mondrian-based heuristics on the Adult database. Adult $\in \mathcal{X}_{48842,3}$

	Runtime in s on Adult					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MONDRIAN	0.11	0.05	0.04	0.04	0.03	0.03
MONDRIAN_V	0.09	0.05	0.05	0.05	0.04	0.04
MONDRIAN_V2D	0.20	0.09	0.07	0.07	0.07	0.07
$\text{MONA}_{0.1}$	0.12	0.05	0.04	0.05	0.03	0.04
$\text{MONA\_2D}_{0.1}$	0.22	0.10	0.07	0.07	0.06	0.06
$\text{MONA}_{0.2}$	0.24	0.06	0.04	0.05	0.04	0.04
$\text{MONA\_2D}_{0.2}$	0.30	0.08	0.11	0.07	0.07	0.07
$\text{MONA}_{0.3}$	0.33	0.20	0.17	0.38	0.41	0.70
$\text{MONA\_2D}_{0.3}$	0.36	0.22	0.20	0.40	0.37	0.69
$\text{MONA}_{0.4}$	0.37	0.27	0.23	0.37	0.43	0.42
$\text{MONA\_2D}_{0.4}$	0.37	0.26	0.26	0.38	0.44	0.52
$\text{MONA}_{0.5}$	0.65	0.42	0.49	0.61	0.76	0.77
$\text{MONA\_2D}_{0.5}$	0.61	0.45	0.52	0.69	0.79	0.72
$\text{MONA}_{0.6}$	1.08	0.73	0.83	1.00	1.07	1.10
$\text{MONA\_2D}_{0.6}$	1.04	0.82	0.98	1.10	1.18	0.99
$\text{MONA}_{0.7}$	4.45	2.86	2.85	2.76	2.47	2.10
$\text{MONA\_2D}_{0.7}$	4.54	3.03	3.14	2.90	2.62	2.16
$\text{MONA}_{0.8}$	9.93	6.01	5.58	5.01	4.39	3.57
$\text{MONA\_2D}_{0.8}$	9.44	6.61	5.99	5.44	4.46	3.74
$\text{MONA}_{0.9}$	51.56	36.14	28.09	22.71	15.83	10.80
$\text{MONA\_2D}_{0.9}$	50.66	35.86	28.13	22.77	15.69	11.65
MONA_{1}	226.28	148.64	116.42	99.75	78.56	52.26

Table C.5: Runtimes in s of Mondrian-based heuristics on the Credit Card database. Credit Card $\in \mathcal{X}_{30000,24}$

	Runtime in s on Credit Card					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MONDRIAN	0.14	0.09	0.06	0.06	0.05	0.05
MONDRIAN_V	0.16	0.07	0.06	0.06	0.07	0.06
MONDRIAN_V2D	1.12	1.18	1.04	1.04	1.02	0.97
$\text{MONA}_{0.1}$	0.18	0.08	0.06	0.06	0.10	0.06
$\text{MONA\_2D}_{0.1}$	1.18	1.15	1.03	1.02	1.01	0.96
$\text{MONA}_{0.2}$	0.28	0.15	0.06	0.06	0.06	0.08
$\text{MONA\_2D}_{0.2}$	1.19	1.27	1.00	1.01	0.99	0.95
$\text{MONA}_{0.3}$	0.34	0.20	0.17	0.13	0.14	0.06
$\text{MONA\_2D}_{0.3}$	1.21	1.14	1.07	0.98	1.03	0.96
$\text{MONA}_{0.4}$	0.42	0.27	0.25	0.25	0.28	0.36
$\text{MONA\_2D}_{0.4}$	1.16	1.10	1.04	1.05	1.08	1.15
$\text{MONA}_{0.5}$	0.49	0.36	0.31	0.33	0.36	0.46
$\text{MONA\_2D}_{0.5}$	1.19	1.10	1.04	1.07	1.11	1.21
$\text{MONA}_{0.6}$	0.99	0.71	0.67	0.64	0.77	0.83
$\text{MONA\_2D}_{0.6}$	1.42	1.35	1.24	1.24	1.28	1.46
$\text{MONA}_{0.7}$	1.64	1.14	1.08	1.06	1.06	1.25
$\text{MONA\_2D}_{0.7}$	1.94	1.79	1.62	1.56	1.62	1.75
$\text{MONA}_{0.8}$	6.14	4.87	4.15	3.67	3.41	3.38
$\text{MONA\_2D}_{0.8}$	6.74	5.02	4.38	4.19	3.72	3.62
$\text{MONA}_{0.9}$	11.79	9.28	8.28	8.13	6.75	5.96
$\text{MONA\_2D}_{0.9}$	12.28	9.59	9.23	7.56	7.02	6.31
ONA*	49.27	42.92	35.37	36.49	32.84	29.62

Table C.6: Runtimes in s of Mondrian-based heuristics on the Winnipeg database. Winnipeg $\in \mathcal{X}_{325834,174}$

	Runtime in s on Winnipeg					
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 7$	$k = 10$
MONDRIAN	6.8	6.4	6.2	5.9	5.8	5.6
MONDRIAN_V	9.4	8.8	8.7	8.7	8.7	8.3
MONDRIAN_V2D	1407.7	1475.3	1483.6	1485.4	1416.4	1323.5
$\text{MONA}_{0.1}$	9.8	8.1	8.0	7.9	7.9	7.7
$\text{MONA\_2D}_{0.1}$	1766.1	1455.2	1487.7	1492.6	1420.8	1402.8
$\text{MONA}_{0.2}$	12.6	11.8	11.5	8.0	8.4	8.1
$\text{MONA\_2D}_{0.2}$	1623.1	1360.2	1399.3	1363.2	1449.1	1392.6
$\text{MONA}_{0.3}$	16.6	16.8	16.5	17.0	20.1	19.3
$\text{MONA\_2D}_{0.3}$	1489.9	1387.4	1342.1	1356.5	1394.2	1396.8
$\text{MONA}_{0.4}$	27.8	28.7	27.9	28.3	30.7	38.2
$\text{MONA\_2D}_{0.4}$	1568.4	1348.1	1249.4	1246.6	1262.6	1294.7
$\text{MONA}_{0.5}$	45.7	42.0	41.0	40.6	42.3	51.0
$\text{MONA\_2D}_{0.5}$	1533.6	1223.9	1239.4	1208.0	1298.1	1285.7
$\text{MONA}_{0.6}$	171.6	149.9	139.5	133.6	122.9	126.6
$\text{MONA\_2D}_{0.6}$	1386.1	1283.0	1300.5	1266.7	1275.2	1246.4
$\text{MONA}_{0.7}$	721.3	646.2	591.8	554.9	495.8	453.6
$\text{MONA\_2D}_{0.7}$	1810.3	1572.3	1597.0	1501.8	1386.3	1354.9
$\text{MONA}_{0.8}$	3573.4	3215.4	2841.7	2753.5	2362.4	2098.5
$\text{MONA\_2D}_{0.8}$	4456.8	3767.2	3554.8	3409.7	3095.9	2789.4

D

Publications

2018

Improving Anonymization Clustering: This conference contribution written in cooperation with Rüdiger Reischuk has been presented at the 2018 edition of the regular convention *SICHERHEIT* of the security division of the German Informatics Society (GI) held in Constance, Germany in April 2018. The paper has been peer-reviewed and appeared in the conference proceedings as part of the LNI (Lecture Notes in Informatics) series [70].
The paper introduces `MDAV*` as a new variant of maximum distance heuristics for the k-anonymous microaggregation problem. It further introduces a preliminary version of the formal definitions of microaggregation presented in chapter 2 of this thesis.

2019

Hardness of k-anonymous microaggregation: This workshop contribution has been presented at the 17th edition of the Cologne-Twente Workshop on Graphs and Combinatorial Optimization held in Enschede, Netherlands in July 2019. The paper has been peer-reviewed and appeared in the workshop proceedings [68].
The paper is an extended abstract of a work-in-progress version of the complexity results presented in section 4.2 of this thesis. It is restricted to cases in which k is part of the input and larger than 25.

2020

Hardness of k-anonymous microaggregation: This journal paper written in cooperation with Rüdiger Reischuk has been published in October 2020 as a peer-reviewed corrected proof of the *Discrete Applied Mathematics* journal [69]. It contains the hardness result presented in section 4.2 in its final form.

2021 *Scalable k-anonymous Microaggregation: Exploiting the Trade-off between Computational Complexity and Information Loss*: This conference contribution written in cooperation with Rüdiger Reischuk has been presented at the 18th International Conference on Security and Cryptography (SECRYPT 2021) in July 2021. The paper introduces `ONA*`, `MONDRIAN_V`, `MONDRIAN_V2D`, `MONA` and `MONA_2D` as presented in chapters 6 and 7. It has been peer-reviewed and appeared in the conference proceedings [71].

E

Supervised Bachelor's and Master's Theses

During my work for this dissertation I supervised several bachelor's and master's theses with topics both related and unrelated to my own research activity.

Related

2018: Mustafa Sahin — *Evaluability preserving database anonymization with the help of current algorithms*:
In his bachelor's thesis Mr. Sahin analyzed several maximum distance heuristics including `MDAV*` by empirical evaluation and further evaluated possible improvements to `MDAV*`.

2019: Finn Christian Stoldt — *Database Anonymization Based on k-means Algorithms*:
In his bachelor's thesis Mr. Stoldt analyzed the existing merge and split approach, evaluated possible improvements by an evolutionary strategy and designed tools to visualize information loss.

2021: Yara Sophie Schütt — *Design and Analysis of Anonymization Algorithms Preserving Evaluability under Consideration of Differential Privacy*:
In her bachelor's thesis Ms. Schütt analyzed the applicability of `SmallDB` and safe k-anonymization for the differentially private release of synthetic data similar to an original database.

Unrelated

2016: Gudrun Mareike Amedick — *Placement of storage nodes in dynamic networks*:
In her bachelor's thesis Ms. Amedick used graph theoretical models to create algorithms finding optimal positions and capacities of storage nodes within idealized power grids.

2018: Ivo Heinecke

Private Edit Distance on DNA:
In his master's thesis Mr. Heinecke designed and analyzed methods to compute fast private edit distance on human DNA. He used techniques that transform the problem into private set operations on sets of genetic markers.