UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Implementation and Evaluation of Microaggregation Algorithms for Categorical Data

---

A Master's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Arnold Veltmann

In partial fulfilment

of the requirements for the degree of

Cybersecurity **ENGINEERING**

Advisor: Esteve Pallarès Segarra, MSc

Barcelona, September 2022

Abstract

In a growingly digitalised world, the need for data privacy is apparent. Data scientists have contributed much previous work to ensure privacy regarding numerical data attributes in published datasets. However, work with categorical data tends to significantly affect the data utility concerning information loss, and less feasible research is available. The thesis aims to describe, implement and compare multiple microaggregation algorithms for categorical data. To achieve the goals of the thesis, and provide valuable output, multiple new proposals to handle categorical data based on the Mondrian algorithm were presented as part of the thesis. It was found that the proposals fared well compared to some previously presented algorithms, both in terms of algorithm execution time and potential information loss and reidentification risk.

Acknowledgements

The help of the advisor, Esteve Pallarès Segarra, in the topic choice, providing related works of literature, and active willingness to discuss different aspects of the thesis and concepts of data privacy, in general, is gratefully acknowledged. In addition, the comments and thoughts of Lemmo Lavonen, who was up to discuss code and theory-related questions concerning prior works whenever any discussion was required, were also valuable for improving the outcome of the thesis.

Table of contents

List of Figures

List of Tables

1. Introduction

As the world is growing increasingly digitalised and individuals' data has become a commodity, the need for privacy on both an individual and institutional level has become more evident. In this principle, statistical disclosure control (SDC) aims to reduce the risk of information disclosure in published datasets. However, not all data is equal, and researchers may use different anonymisation methods based on the data's distinct characteristics. Furthermore, there are multiple ways of dividing data into categories. One common recourse is to classify the data as continuous or categorical.

Continuous. An attribute is continuous if it is numerical, and arithmetical operations (such as finding the mean) can be performed on it [1]. A person's *income* and *net worth* can be considered continuous attributes.

Categorical. An attribute is categorical when it takes values over a finite set, and standard arithmetical operations cannot effectively be performed on it [*Ibid.*]. A person's *sex* and *address* can be considered categorical attributes.

A prevalent way to reduce the risk of information disclosure is to use microaggregation algorithms on the data before release, which reduces the data utility but improves the level of data privacy. While microaggregation is most often used in conjunction with continuous data, it is possible to adapt the data masking algorithms for categorical microdata [3].

Frequently, microaggregation algorithms aim to provide *k*-anonymity. The concept of *k*-anonymity is well-spread in research related to data anonymisation. It was introduced in 1998 by Latanya Sweeney and Pierangela Samarati [2] to reduce the risks of re-identification in published data. In the simplest sense, a set of published records is *k*-anonymous if for every record exists at least $k - 1$ additional records indistinguishable from the record under observation. Typically, the first step in a *k*-anonymity guaranteeing microaggregation algorithm is to form data clusters containing at minimum *k* elements. In the second step, the average for each cluster is calculated, and the original data is replaced with the averages [3].

Examples of popular microaggregation algorithms to obtain *k*-anonymity in a dataset include the Maximum Distance to Average Vector (MDAV) and the Mondrian algorithms.

1.1. Statement and purpose

As stated, microaggregation is most associated with continuous data. However, the thesis focuses on microaggregation algorithms for categorical data, wherein less research is available. The aim of the thesis is to describe, implement and compare multiple different microaggregation algorithms in relation to categorical attributes.

To achieve the thesis's goals and provide a relatively fair background for comparison, the algorithms are programmed from scratch to have a structurally similar codebase. Python was chosen as the programming language to implement the algorithms. More specifically, the thesis is to provide comparisons regarding information loss, reidentification risk, and the execution time measurements of the algorithms. However, it is essential to note that runtime comparisons heavily depend on the programming language and the use of different data structures and libraries. Thus, it serves as an indicator of the potential and is not such a definite comparison factor as the information loss and risk measurements.

The produced work hopes to create a further basis for research in the future and highlight some of the difficulties of working with categorical data.

1.2. Methods and procedures

Microaggregation with categorical data has been researched only by a limited circle of researchers to the best of the author’s knowledge.

The algorithms under observation in the thesis are modified variants of the Maximum Distance to Average Vector (MDAV) and Mondrian algorithms, which have been adapted for use with categorical data. The MDAV adaptations are implementations based on the proposals of Sánchez et al. [6] in 2012. The specific Mondrian adaptations are introduced in this thesis. Wherein, one of the Mondrian adaptations remains relatively faithful to the strict multidimensional Mondrian algorithm described by LeFevre et al. [7] in 2006, and the MDAV adaptation proposals in the work of Sánchez et al. inspire the second [6].

There are multiple prior papers on microaggregation with categorical data, e.g., the ground-laying work of Domingo-Ferrer et al. in 2005 [1] and later the more closely related work of Abril et al. in 2010 [8], highlighting the importance of semantics. However, SA-MDAV has proved to be one of the more feasible proposals for microaggregation with categorical data, as works further in the past have introduced quite heavy loss of data utility [6]. Despite this,

no prior functional code is available publicly for the SA-MDAV algorithm, which further prompts an implementation of the algorithm from scratch.

In that regard, the primary comparison metric for information loss in this work is the same as proposed in the paper by Sánchez et al. [*Ibid.*]. Using the same metric and related terminology (such as the Sum of Squares Error (SSE) and Wu-Palmer distance) allows easy and understandable comparisons between the algorithms and papers. The concepts above and additional topic-specific terminology and background information are explained in the thesis's second chapter, providing the solid groundwork to describe the rest of the work and results. The third chapter describes the tasks completed regarding specific datasets used to gather results and the adaption of the Mondrian algorithm for categorical data. In chapter four, the results are visualised and analysed. The final chapter of the thesis concludes the work and discusses future research possibilities.

1.3. Work plan

The author worked on the thesis from February to August 2022. Before gathering results, extensive research into the topic was carried out to determine, which algorithms to compare, and how to compare them. The algorithms were further researched and then implemented as functional code. A Gantt diagram outlining the rough schedule of the project is visualised in Figure 1.

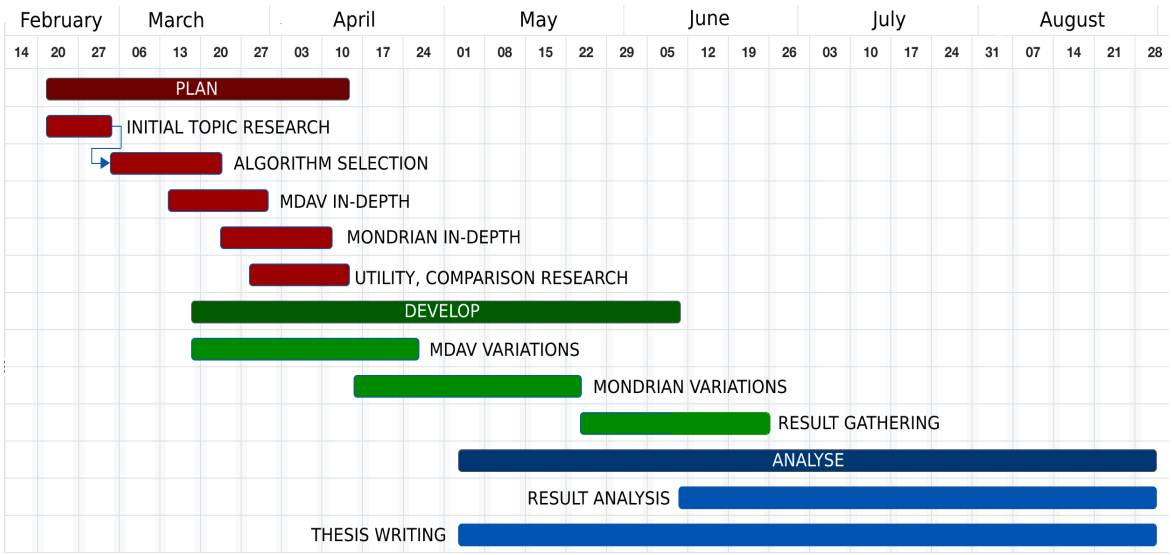


Figure 1. Gantt diagram of the thesis schedule

2. State of the art

Chapter *2.1.* provides the basic background information concerning terminology and concepts to lay the foundation for a better understanding of the thesis.

The following chapter *2.2.* serves to provide an overview of previous work and studies (as well as the descriptions of the semantics based MDAV algorithms) related to the topic, based on which it is possible to create a more extensive foundation for the presentation of ideas, confirmation of results, and any related discussion.

2.1. Background

The following chapters *(2.1.1. - 2.1.5.)* describe some of the key concepts and terminology which the thesis is based upon.

2.1.1. Value Generalisation Hierarchies

A value generalisation hierarchy (VGH) is an organised grouping of entities based on the common attributes that they share. In other words, in a generalisation hierarchy, a superclass (or supertype) is connected with one or more subclasses (or subtypes), wherein the subclass is a specialisation of the superclass. A multilevel hierarchical tree is formed in doing so, as a subclass can have its own subclasses [10].

These hierarchies can either consist of numerical or categorical data. For example, in numerical attribute VGH, the range 1 to 100 could be split into two subclasses: 1 to 50 and 51 to 100. The output is a two-level value generalisation hierarchy. A categorical attribute can be divided into subclasses based on any underlying ontology. An example of a two-level categorical attribute VGH is visualised in Figure 2.

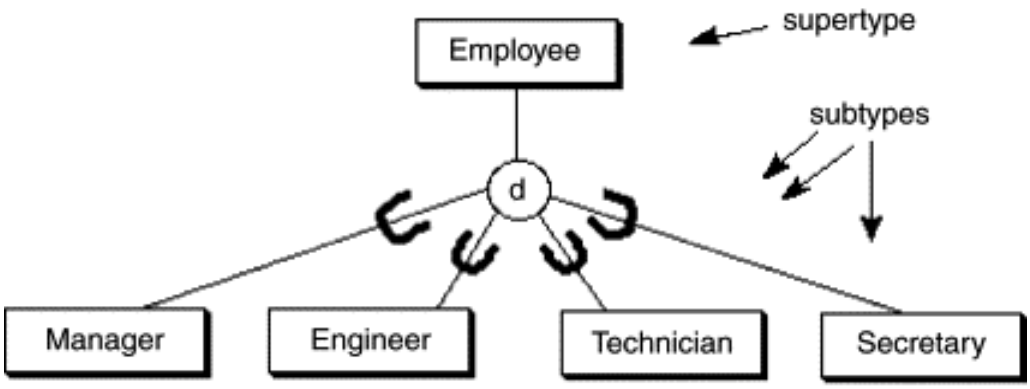


Figure 2. Two-level categorical attribute Value Generalisation Hierarchy [9]

In the figure, the visualised *d* between the super and subtypes specifies disjointness, which means that the entities are mutually exclusive [*Ibid.*]. The *employee* is the highest-level abstraction and is the superclass, while other, more specialised, values are its subclasses.

2.1.2. WordNet

As stated in the previous chapter, a value generalisation hierarchy based on a categorical attribute is best built on some background ontology.

WordNet is a lexical database of English, where words are grouped into sets (Synsets), which are interlinked by utilising conceptual-semantic and lexical relations. Princeton University owns the trademark for WordNet, yet it is available for research and commercial use free of charge [4].

In the way it is structured, WordNet can be used as a lexical ontology to create unbiased Value Generalisation Hierarchies. With a standardised background ontology, related works can also be better compared. In this thesis, WordNet 3.1 was used.

2.1.3. Wu-Palmer similarity

Wu-Palmer similarity has been used in previous related works (such as by Sánchez et al. [6] and Abril et al. [8]) as a measure to compare the similarity of categorical attributes (or in the case of WordNet, Synsets) with each other.

Wu and Palmer [5] define the conceptual similarity between two concepts *C1* and *C2*, as:

$$ConSim(C1, C2) = \frac{2 * \mathrm{N3}}{\mathrm{N1} + \mathrm{N2} + 2 * \mathrm{N3}}$$

Where *N3* denotes the depth of the Least Common Subsumer (LCS) of *C1* and *C2* in the VGH. *N1* and *N2* denote the depth distance from the LCS to the nodes under comparison.

A more intuitive definition could be:

$$ConSim(C1, C2) = 2 * \frac{\mathrm{N3}}{\mathrm{N1} + \mathrm{N2}}$$

Where N3 remains the same, *N1* and *N2* denote the depth distance from the root, yielding the same result as the previous formula. The result ranges from 1 to 0, marking the spectrum's ends as identical and with no linkage whatsoever respectively.

As per Sánchez et al. [6], the distance between two concepts is derived from the Wu-Palmer Similarity formula as:

$$distance(C1, C2) = 1 - ConSim(\mathrm{C1}, \mathrm{C2})$$

2.1.4. Sum of Squares error

One standard way to evaluate the information loss in a dataset after applying a microaggregation algorithm is through the Sum of Squares Error (SSE).

In terms of data loss, the optimal *k*-partition maximizes within-group homogeneity, as microaggregation replaces the values in the cluster by the given cluster's centroid (often the mean or mode). The algorithm for calculating SSE is defined as follows [3]:

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \overline{x}_i)' (x_{ij} - \overline{x}_i)$$

For each record in the original dataset, the numerical difference (most commonly Euclidean distance) between the original and the microaggregated records is squared and summed together. The operation is performed on normalised values.

However, as the input and output are both numerical, a different formula must be considered when calculating the Sum of Squares Error for categorical data.

In this thesis, instead of the commonly used Euclidean distance in determining the distance between two datapoints, the distance measure based on Wu Palmer similarity proposed by Sánchez et al. [6] is used and is explained in the previous paragraph (2.1.3).

2.2. Previous related work

This paragraph gives a brief overview of some previous works regarding microaggregation and microaggregation specifically in conjunction with categorical data.

2.2.1. *K*-modes

One of the earliest microaggregation algorithms proposed (in 2004) regarding categorical data is the *k*-modes algorithm [14], which is based on the popular *k*-means algorithm for numerical data.

In the *k*-modes algorithm, the process is started by choosing an initial partition or cluster at random. Next, the number of dissimilarities is calculated for each record in the dataset. For nominal data – the distance is defined as 1 in case values are different and 0 when they are equal. In case of ordinal data, the distance is calculated arithmetically considering the spacing of the attributes in the ordered data. In the third step, new modes (most observed values) are calculated and chosen as representatives. These steps, starting from the second, are repeated to form the clusters [*Ibid.*].

2.2.2. MDAV-generic

In 2005, Domingo-Ferrer et al. proposed MDAV-generic [1], which compared to the prior works with MDAV, made it fit for the first time to categorical data.

Like the k-modes algorithm (see chapter 2.2.1), the distance between two nominal attributes was defined using the equality predicate – counting the value between two nominal attributes as 0 in case of inequality and 1 in case of equality. To calculate the averages, likewise, the mode was used. However, the two algorithms differ in partitioning, as the MDAV-generic algorithm does not require to define the number of clusters in the final output as an input [*Ibid.*].

2.2.3. Semantic and Semantic-Adaptive MDAV

In their paper, Sánchez et al. [6] introduced two new algorithms – Semantic MDAV and Semantic Adaptive MDAV, wherein the first is in a way a layer for the second one.

In the proposed algorithms, emphasis is put on data semantics, as it is argued that the simplistic treatment of categorical data in prior works significantly contributes to data loss. Thus, in the algorithms, a structured knowledge source such as WordNet (however a different background ontology could be used as well) is used to map the distances between concepts, while previous works rely on the equality predicate. Based on the created VGH, Weighted Semantic Distance (see 2.2.3.1) is used to calculate the similarity of concepts in the clustering process and in the later steps of loss evaluation [*Ibid.*].

The Semantic Adaptive MDAV uses the same underlying concepts and logic as the Semantic MDAV algorithm. However, the adaptive part in the name refers to the fact that the resulting clusters do not have to contain *k* records; instead, they must contain, at minimum, *k* records. It is argued that due to the discrete nature of categorical data, it is more

desirable in terms of cohesion to include all records with identical values in the same cluster. Thus, the *k-anonymity* property is met, while the cluster size can vary depending on the data distribution. In addition, for the algorithms, the centroid is recalculated each time there is a change in the data, in comparison to the classical MDAV algorithm [*Ibid.*]. The data is pre-processed in a way that equivalent records are mapped by frequency, which contributes greatly to the speed of the algorithm.

The following complete pseudocode [*Ibid.*] of the SA-MDAV algorithm is taken from the SA-MDAV paper, with some slight changes, which should not contribute to any differences in the results. Namely, creating an empty array to store the created clusters instead of creating a copy of the original dataset for continuous modification. The purpose of the modification is to give an accurate representation of the work done regarding the implementation of the algorithm for this thesis (Appendix B).

ALGORITHM 1. SEMANTIC ADAPTIVE MAXIMUM DISTANCE TO AVERAGE VECTOR

Input: D (dataset), k (level of anonymity)
Output: CD (data divided into clusters)

```
1   CD ← new empty array
2   while (|D| ≥ k) do
3       Compute the centroid x of all tuples in D
4       Consider the most distant tuple r to the centroid x
5       Form a cluster C with the tuple r and remove this tuple from D
6       Calculate centroid c
7       while (|C| < k) do
8           Find the closest tuple t to c from the tuples in D and add it to C
9           Remove the tuple t from D
10          Calculate the new centroid c of cluster C
11      end while
12      Add cluster C to CD
13      if (|D| ≥ k) then
14          Find the most distant tuple s to tuple r
15          Form a cluster C with the tuple s. Calculate centroid c
16          while (|C| < k) do
17              Find the closest tuple t to c from the tuples in D and add it to C
18              Remove the tuple t from D
```

```
            Calculate the new centroid c of cluster C
        end while
        Add cluster C to CD
    end if
end while
Add each remaining tuple in D to their closest cluster in CD
return CD
```

2.2.3.1. Weighted Semantic Distance

Sánchez et al. [*Ibid.*] propose a distance measure based on Wu Palmer similarity and distance (see 2.1.3). However, theoretically, the Semantic MDAV algorithm could be implemented with a different distance measure.

The following equation [*Ibid.*] describes the weighted semantic distance between two data entries, where v_1 and v_2 describe the attribute's values and ω_1 and ω_2 their respective frequencies. As a clarification, in the case of S-MDAV, the frequencies when calculating the distance are equal to 1, as each record is considered a unique data entry. The weighted semantic distance between the two entries is equal to the Wu Palmer distance for the values times the multiplication of the respective frequencies:

$$wsd(< v_1, \omega_1 >, < v_2, \omega_2 >) = \sum_{i=1}^{\omega_1} \sum_{j=1}^{\omega_2} dis_{w\&p}(v_1, v_2) = dis_{w\&p}(v_1, v_2) \times (\omega_1 \times \omega_2)$$

For multivariate data entries, the distances for each corresponding attribute value are added up and then divided by the count of attributes under observation *m*, as can be seen in the following equation [*Ibid.*]:

$$wsd(< \{v_{11}, .., v_{1m}\}, \omega_1 >, < v_{21}, \dots, v_{2m}\}, \omega_2 >) = \sum_{j=1}^{m} \frac{wsd(< v_{1j}, \omega_1 >, < v_{2j}, \omega_2 >)}{m}$$

2.2.3.2. Centroid construction

In the proposal [*Ibid.*], the centroid of a cluster is selected according to the semantics of data, rather than its distribution.

The paper highlights two important differences in calculating the centroid in comparison to previous works:

a) The notion of distance is considered during the centroid construction.
b) The centroid candidates are not limited to the values in the cluster. Instead, they can also be any taxonomical ancestor in the created VGH, allowing the creation of more accurate centroids.

Thus, a centroid is chosen such that the concept minimises the weighted semantic distance concerning all the values in the cluster. Or, in more depth, for each node in the VGH, the distance of the corresponding (matching attribute) value in the cluster is calculated. The node with the shortest distance to all cluster elements is chosen as the centroid for the cluster. The distance measures both for univariate and multivariate inputs are described in chapter 2.2.3.1. Note that during the centroid calculation, the input frequency of the node in the VGH is equal to 1.

2.2.3.3. Calculating loss

The loss is calculated using the Sum of Squares error, described in chapter 2.1.4 of the thesis. However, as stated in the chapter, Euclidean distance is most often used to calculate the SSE, which is not applicable for categorical values.

Sánchez et al. [*Ibid.*] use the weighted semantic distance measure based on Wu Palmer distance described in 2.1.3 instead in the SSE equation:

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (dis_{w\&p}(x_{ij}, \overline{x}_i))^2$$

Wherein the SSE of the dataset is equal to the sum of (squared) distances of each entry in each cluster to the cluster centroid. Note that in theory, as the values are pre-processed, the Wu Palmer measure translates here as the distance measure explained in 2.2.3.1, since the SSE must be calculated in relation to each record.

Furthermore, for loss calculation, it is also necessary to calculate the Sum of Squares Total (SST). The measure calculates the sum of (squared) distances between each element and the centroid of the entire dataset.

After calculating the values, the loss (in percentage) can be calculated as

$$L = \frac{SSE}{SST} \times 100$$

2.2.4. Mondrian

Mondrian is considered a recursive algorithm and is classically used for numerical, ordinal data. Due to its recursive factor, it is best visualised with a graph (see Figure 3).

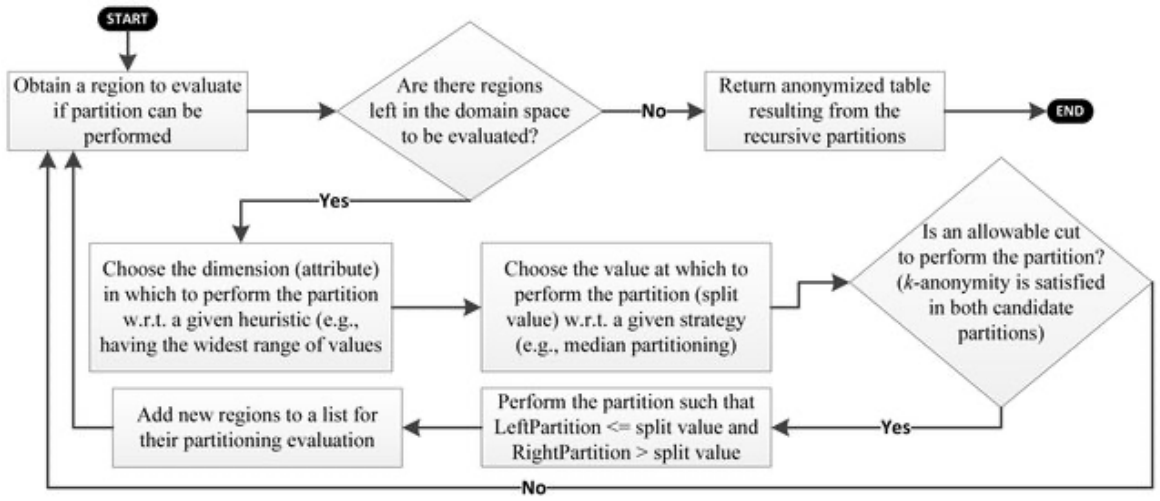


Figure 3. Mondrian algorithm [11]

There are multiple papers that include pseudocode for the Mondrian algorithm, however, for clarity, a simplified pseudocode demonstrating the core of the algorithm is described below (Algorithm 2), which is roughly based on the code map visualised in Figure 3.

ALGORITHM 2. MONDRIAN CORE

Input: D (dataset), k (level of anonymity)
Output: CD (data divided into clusters)

```
1   CD ← new empty array
2   Call recursive algorithm on the entire dataset: Mondrian(D)
3       if (no allowable cut for input data) then
4           Create cluster C from D
5           Add cluster C to CD
6           return
7       else
8           Choose dimension (attribute) A to split
9           Evaluate the split point (e.g., median) for A and find its index i
```

10 Split D based on index i to LeftPartition and RightPartition
11 *Mondrian*(LeftPartition)
12 *Mondrian*(RightPartition)
13 Output CD

A more detailed pseudocode in relation to the thesis can be found in the third chapter of the thesis (*3.2*), outlying some of the more specific approaches.

Mondrian can be modified and classified by different means, such as being multidimensional or single-dimensional and strict or relaxed, as described by LeFevre et al. in 2006. In their paper, it was concluded that strict multidimensional partitioning provided the best results regarding data utility. Furthermore, despite Mondrian being a greedy algorithm, it often produced better results compared to more expensive optimal algorithms for other recoding models [7]. Note that already at its core, strict-Mondrian aims to provide output data that satisfies *k-anonymity*, but does not restrict clusters to *k* values, and thus already possesses the adaptability property described in the proposal of SA-MDAV [6] wherein equal values are highly likely to end in the same cluster.

Due to the reasons above, the Mondrian implementations in this thesis are also based on strict multidimensional partitioning. As per the name, the multidimensional part suggests that the data can be split in different dimensions, while the strict part suggests that no split is made such that a dimension is split in a way that equal values could end up in both the left and right partition. A visual representation of the partitioning for numerical data is presented in Figure 4.

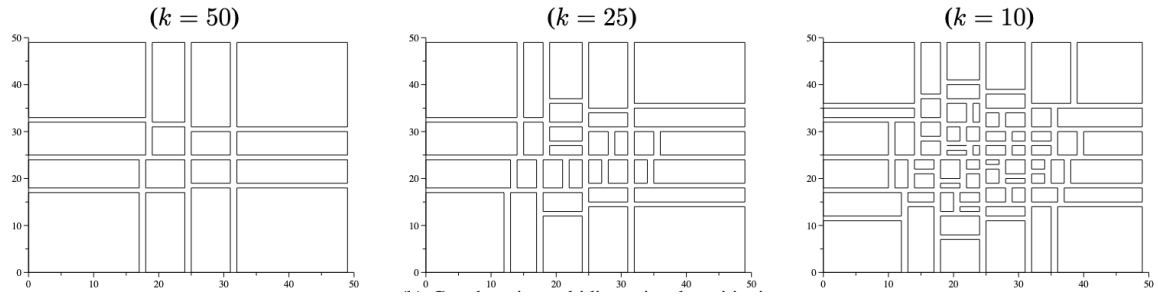


Figure 4. Multidimensional strict partitioning [7]

2.2.5. Mondrian for categorical data

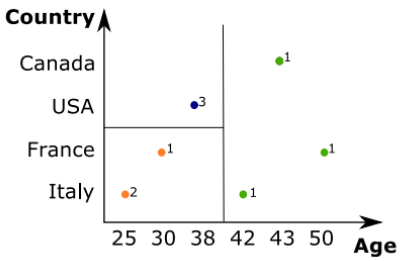
While likewise to MDAV, most work regarding Mondrian is done with numerical data, the transformation to use Mondrian with categorical data is in theory simpler. However, it does

garner some difficulty if data semantics are of great importance. To adjust the standard Mondrian algorithm for categorical data, it is necessary to describe a way to order the categorical values. To create cohesive clusters, the most similar values should intuitively be ordered close to each other.

One such proposal is a part of the EU-funded MOSAICrOWN projects' [15] specifications, which describe a mixed-data Mondrian algorithm. The proposed algorithm is based on the multi-dimensional Mondrian algorithm. In the project, the categorical attributes are ordered based on their appearance in the leaf level of the created Value Generalisation Hierarchy. In contrast to other works, instead of replacing the values in a cluster with their averages or mean values, the Least Common Subsumer is chosen as the recoded value for the cluster elements [*Ibid.*]. This means, for example, that if a cluster consists of values {'*France*', '*Italy*', '*Italy*'} and k is equal to 3, all the values in the cluster will be recoded as *Europe* instead of *Italy*. The algorithm process is visualised in Figure 5 below, where a) is the original data, b) shows the partitioning, and c) shows the recoded data after partitioning and generalisation.

Age	Country	TopSpeed
25	Italy	132
25	Italy	132
30	France	128
42	Italy	110
50	France	115
43	Canada	115
38	USA	126
38	USA	127
38	USA	140

(a)



(b)

Age	Country	TopSpeed
[25,30]	Europe	132
[25,30]	Europe	132
[25,30]	Europe	128
[42,50]	World	110
[42,50]	World	115
[42,50]	World	115
38	USA	126
38	USA	127
38	USA	140

(c)

Figure 5. Algorithm process visualisation for mixed-value Mondrian [Ibid.]

The work does not elaborate on how the ordering can be adjusted to deal with cases where not all values in the dataset are placed in the leaf nodes, and some might already be generalisations and thus reside in higher levels of the VGH. As per the algorithm description, the dimension is split at the median value. This, presumably based on the illustrations, means that the cut takes place for categorical values such that the left and right partitions contain an equal number of records.

3. Project development

In this paragraph, the work done in order to obtain the results for comparison and analysis is described. Starting with the descriptions of the datasets used, Value Generalisation Hierarchies created, and within containing a brief overview of some of the issues faced and decisions taken. Finally, more specific work in implementing the Mondrian and MDAV algorithms is discussed.

3.1. Used datasets and VGHs

As stated in 3.1.2, WordNet can be treated as a background ontology when creating VGHs for categorical data. However, this introduces some limitations, as not all concepts can be mapped as Synsets within WordNet. For example, creating a VGH based on WordNet for an attribute describing *operating systems* is impossible. In this scenario, a different ontology must be used or created.

In this thesis, two datasets were used, wherein one was further divided into smaller datasets to obtain more different results for comparison. For the first dataset, VGHs based on WordNet with slight modifications were created, while due to the nature of the attributes and their properties, for the second dataset, VGHs based on the authors' general knowledge were created. All used datasets are included as part of Appendix A.

3.1.1. Adult Census dataset

The first dataset is the publicly available *Adult Census* dataset [12] in the UCI repository, which has previously been used in other works such as by Sánchez et al. [6] and prior works.

Similar to the paper mentioned above, some replacements had to be made to the dataset due to some of the modalities in the data not being directly found in WordNet. Thus, the changes were required to create proper VGHs and map the distances for each term used in the data. Furthermore, likewise to past work by Sánchez et al., two categorical attributes from the dataset were used – *occupation* and *native.country*.

The *occupation* attribute

Multiple transformations were required to be performed on the data before a Value Generalisation Hierarchy based on the occupation attribute could be constructed. These value transformations are visualised in Table 1.

Original value	*Used WordNet Synset*
Tech-support	technician
Craft-repair	craftsman
Other-service	worker
Sales	salesperson
Exec-managerial	executive
Prof-specialty	specialist
Handlers-cleaners	cleaner *(sense 3)*
Machine-op-inspct	machinist
Adm-clerical	clerk
Farming-fishing	farmer
Transport-moving	mover
Priv-house-serv	housekeeper
Protective-serv	guard
Armed-Forces	soldier

Table 1. Transformations in the occupation attribute to adapt for WordNet

Most of the replacements were the same as those by Sánchez et al. [*Ibid*.], with a few exceptions. It was not explicitly clear in the paper mentioned above whether the built VGH reached out to the root of the WordNet ontology or not, for clarity, the VGH in this thesis reached to the root, *entity*. Furthermore, in contrast to the work by Sánchez et al., an additional replacement of *sales* with *salesperson* took place. Moreover, *machine-op-inspct* was replaced by *machinist* in favour of *operator; farming-fishing* was replaced with *farmer* in favour of a more general *skilled_worker;* and *transport-moving* was replaced with *mover* in favour of *carrier*.

For all Synsets, the sense 1 of the word was used when creating the VGH, except for *cleaner*, where sense 3 seemed more fitting for the concept of an *occupation* based VGH, as in WordNet 3.1 [4] sense 1 referred to a cleanser.

Regarding the occupation attribute, a Value Generalisation Hierarchy based on the WordNet ontology is visualised in Figure 6.

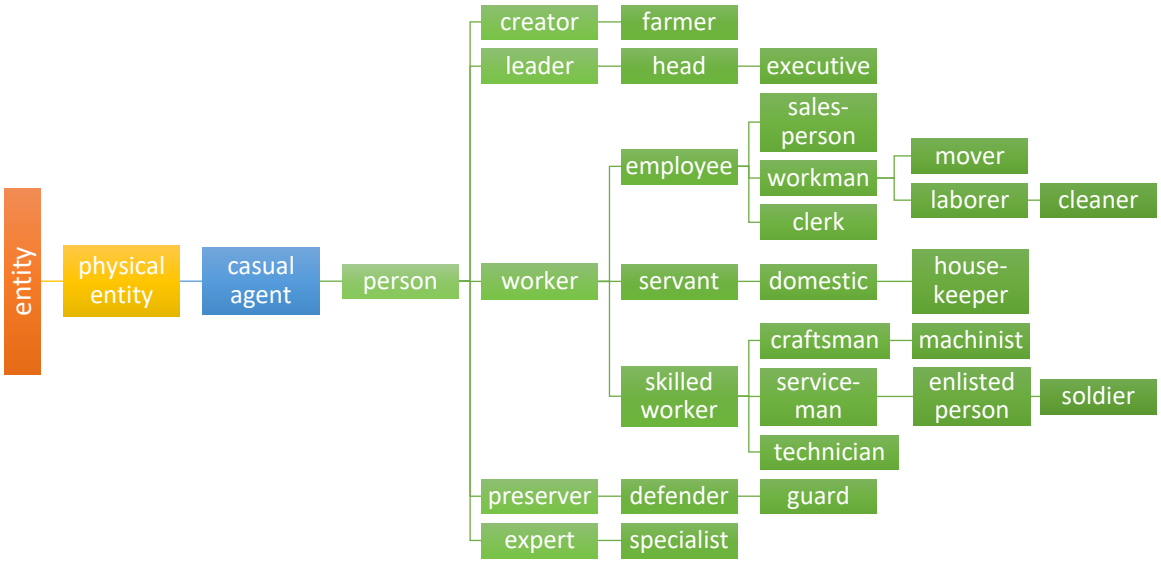


Figure 6. Value Generalisation Hierarchy of the occupation attribute (WordNet)

The *native.country* attribute

In the case of the *native.country* attribute, fewer substitutions were required. These substitutions were performed as by Sánchez et al. [6.] and are visualised in Table 2.

Original value	*Used WordNet Synset*
Outlying-US(GUAM-USVI-etc)	american_state
Hong	hong_kong
Holand-Netherlands	netherlands

Table 2. Transformations in the native.country attribute to adapt for WordNet

However, outside of the WordNet based ontology, a more appropriate or precise substitution might be considered for replacing the *Outlying-US(GUAM-USVI-etc)* value.

The VGH for the *native.country* attribute was created based upon WordNet ontology as prior; however, the liberty to distance from the WordNet based ontology was taken in the case of one single variable - *Taiwan.* It was moved in the VGH to be classified among other Asian countries, as compared to being considered an *island* and building a distinct separate branch for it in the Value Generalisation Hierarchy. For clarification, in the WordNet Lexical Database, *Taiwan* is not considered a country [4].

Furthermore, this highlights why it might make sense to create purpose-built ontologies to build the VGH upon in some circumstances. Further characterised by the fact that with

WordNet, some specifications can go missing. For example, *England* and *Scotland* are not considered closer in the general ontology compared to *England* and *Netherlands* [*Ibid.*]. A more specific VGH can potentially decrease relative data loss significantly. For reproducibility, the created VGH has been visualised and shown in Figure 7.

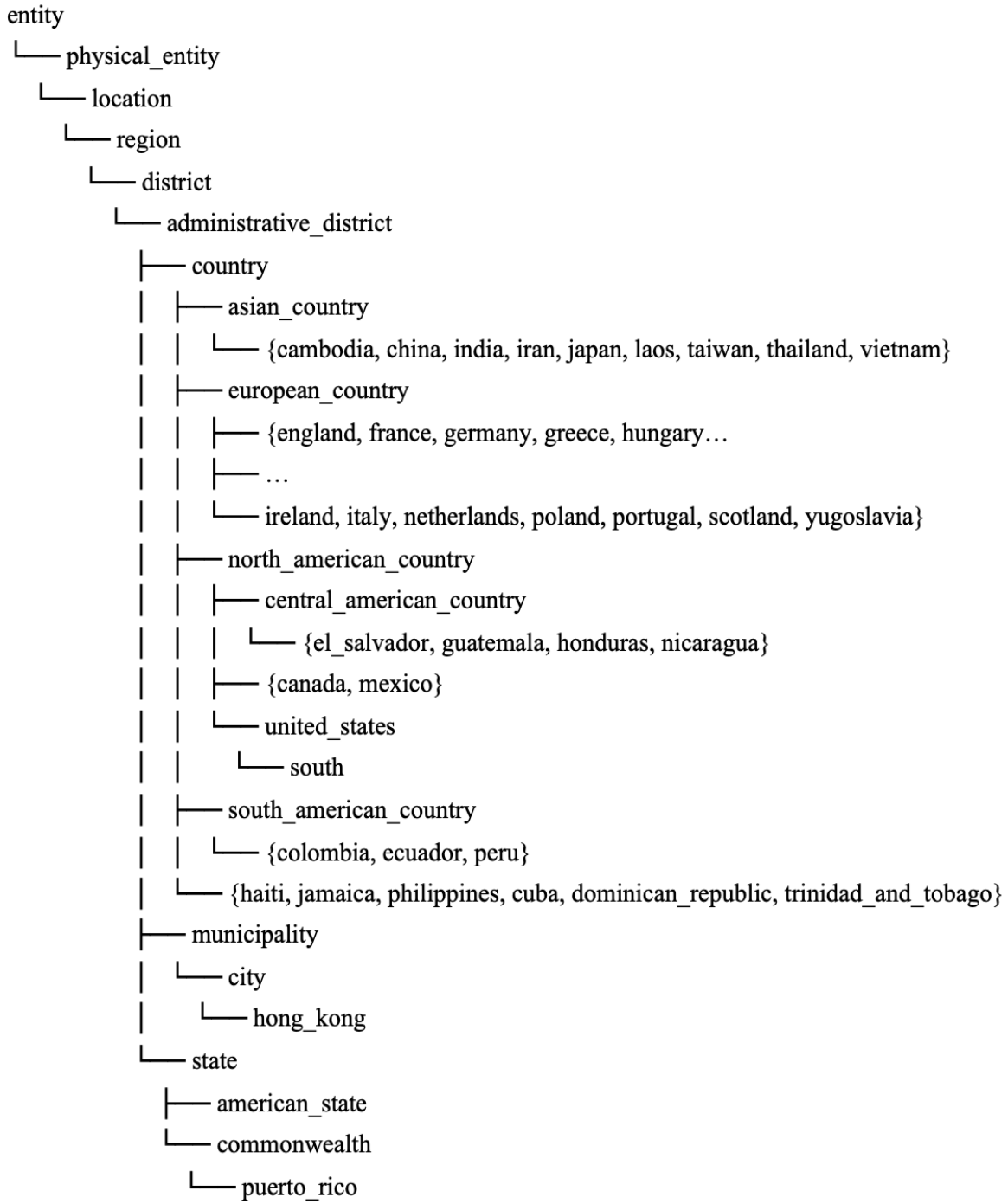


Figure 7. Value Generalisation Hierarchy of the native.country attribute (WordNet)

Notice that one might also consider reconstructing the VGH on different aspects, such as making it more region-specific by defining a separate branch for Caribbean countries or moving *american_state* under the *United States*, *hong_kong* under *China*, and *commonwealth* under *England*, for example.

Data overview

The *Adult Census* dataset consists of 30162 records (after removing n/a values). In the dataset, 84 records (0.278%) violate 2-anonymity and 497 records (1.648%) violate 5-anonymity. The most common value in the dataset corresponds to (‘*executive’*, *‘united_states’*), which makes up 12.38% of the records. The top 5 most frequent values comprise almost half of the entire dataset (48%) and are visualised in Table 3.

occupation	native.country	Freq
executive	united_states	3735
specialist	united_states	3693
craftsman	united_states	3685
clerk	united_states	3449
salesperson	united_states	3364

Table 3. Most frequent values in the Adult Census dataset

The *native.country* attribute contains 41 distinct values, where a whopping 27504 records correspond to *united_states* (91.2%). The *occupation* attribute contains 14 distinct values and is more distributed, where specialist corresponds to 13.9% of all the records.

3.1.2. Rsquared Academy Analytics dataset

The second publicly available dataset has been taken from the Rsquared Academy *Handling Categorical Data in R* series [13].

In this dataset, two categorical attributes were chosen as well – *os* and *country*. However, in contrast, for the *os* attribute, no existing background ontology was used, and for the *country* attribute, the VGH was more loosely based on WordNet and was not built to the root *entity*. This was done due to the fact that, in WordNet, there are no existing concepts to create a VGH for operating systems. The *country* attribute could have been strictly based on WordNet; however, the creative freedom was taken to allow a more specialised VGH

and keep the VGHs structurally similar. In Figure 9, the Value Generalisation Hierarchy created for the *os* attribute is visualised.

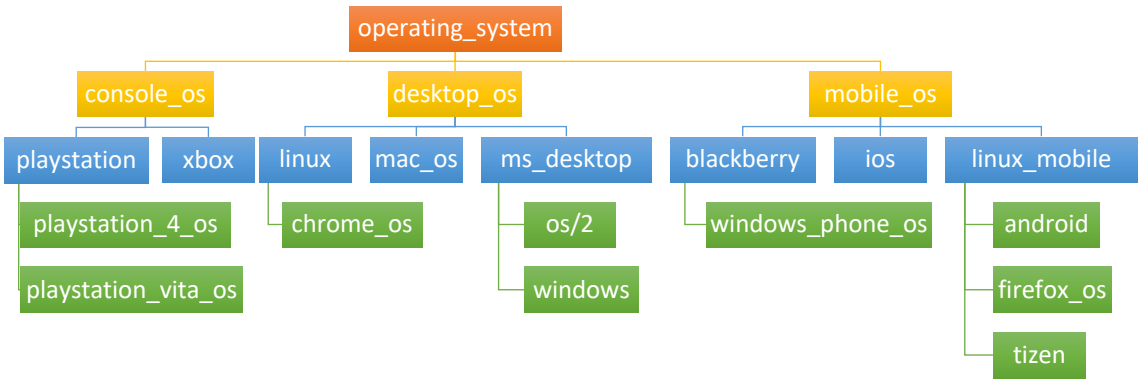


Figure 8. Value Generalisation Hierarchy of the os attribute

As the country attribute consists of 193 unique values, the VGH is not visualised in the thesis; however, it is recreatable with the code attached in Appendix B. As mentioned, the VGH is based loosely on WordNet. However, more specific generalisations were created and *country* was used as the root of the VGH. Some of these generalisations include but are not limited to *oceanian_country*, *carribean_country,* and *balkan_country*.

Based on the *Analytics* dataset, two smaller datasets with 10000 random records each were created to further evaluate the algorithms under comparison in the thesis. These datasets are later in the work referred to as *Analytics Sample One*, and *Analytics Sample Two*.

Data overview

The *Analytics* dataset consists of 243545 records. In the dataset, only 125 records (0.051%) violate 2-anonymity and 531 records (0.218%) violate 5-anonymity. The most common value in the dataset corresponds to (*'windows'*, '*united_states'*), which makes up 13.3% of the records. The top 5 most frequent values comprise a considerable chunk of the entire dataset (39.9%) and are visualised in Table 4.

os	country	Freq
windows	united_states	32390
mac_os	united_states	25341
android	united_states	17691
ios	united_states	10325
chrome_os	united_states	10180

Table 4. Most frequent values in the Analytics dataset

The *os* attribute contains 14 distinct values, where the most frequent value is *windows* with 91208 (37.5%) corresponding records. The *country* attribute contains 193 unique values. Out of all the records, 39.8% of them correspond to *united_states*.

Overview of *Analytics Sample One* and *Analytics Sample Two*

As the datasets are random samples of the *Analytics* dataset, their percentual values regarding most of the values remain similar. However, the smaller datasets allow for obtaining faster results, and possess some distinctness regarding the removed values.

The first dataset violates 2-anonymity on 115 occasions (1.150%) and 5-anonymity on 340 occasions (3.4%). Likewise, to the original dataset, the most common tuple is (*'windows'*, '*united_states'*) with 1354 matching records (13.5%). For the *country* attribute, only 124 distinct values remain, and for the *os* attribute, only 7. In complete, there are 380 distinct value tuples in the formed dataset.

The second dataset is similar, and consist of 396 distinct value tuples. The dataset violates 2-anonymity on 136 occasions (1.36%) and 5-anonymity on 350 (3.5%) occasions. The most common tuple is again (*'windows'*, '*united_states'*) with 1377 occurrences (13.8%). For the *country* and *os* attributes, 129 and 8 distinct tuples remain respectively.

3.2. Mondrian proposals

In this chapter of the thesis, two proposals for adapting Mondrian with categorical data are described. The first is a more-or-less standard approach to the issue, while the second builds upon the first and introduces new logic in determining the split value. Both proposals use the same core algorithm; however, the differences are found in data pre-processing and how the dimensions are split. The pseudocode in Algorithm 3 demonstrates the core of the

Mondrian algorithm in more detail and shows the more realistic view on how the algorithms in Appendix B are programmed in Python.

ALGORITHM 3. MONDRIAN DETAILED

Input: D (dataset), k (level of anonymity)
Output: CD (data divided into clusters)

```
1   CD ← new empty array
2   Call recursive algorithm on the entire dataset: Mondrian(D)
3       Split input D into dimensions and sort by widest range of values
4       if (no allowable cut for input data) then
5           Create cluster C from D
6           Add cluster C to CD
7           return
8       else
9           Choose dimension (attribute) A to split
10          Evaluate the split point for A and find its index i
11          Split D based on index i to leftPartition and rightPartition
12          if (|leftPartition| == k) then
13              Create cluster C from leftPartition and add to CD
14              Mondrian(rightPartition)
15          elif (|rightPartition| == k) then
16              Create cluster C from rightPartition and add to CD
17              Mondrian(leftPartition)
18          else
19              Mondrian(leftPartition)
20              Mondrian(rightPartition)
21  Output CD
```

3.2.1. Multidimensional Strict Mondrian adaptation proposal (Simple Mondrian)

This thesis proposes a way to adapt categorical data for use with the multidimensional strict Mondrian algorithm. The proposal is similar to the categorical adaption described in chapter 2.2.5 of the thesis and remains true to the underlying multidimensional strict Mondrian algorithm demonstrated by LeFevre et al. [7]. While the code produced in this thesis only handles categorical data, then the algorithm and implementation can be easily modified to

handle mixed data, which is, in terms of information loss, significantly more complicated with MDAV implementations.

The main differences between the proposal in this paper, and the algorithm described in 2.2.5 and others are the following:

a) The centroid values can be any existing values in the VGH and are not limited to the Least Common Subsumers of the values inside the cluster, as the cluster centroid calculation is the same as used by Sánchez et al. [6] in their proposal for SA-MDAV and described in chapter 2.2.3.2 of the thesis.
b) The input data can contain values that are generalisations of other existing values within the attribute. In other words, in the created Value Generalisation Hierarchy, the parent nodes can also correspond to some values in the data.
c) The value at which to split the dimension (attribute) is not chosen, such as to have equal left and right partitions, and is not explicitly linked to the mode either; rather, it is determined based on the centroid value of the dimension.

The feature described in b) is especially important in terms of our input data in the *Adult Census* and *Analytics* datasets (see chapters 3.1.1 and 3.1.2 respectively), where such instances occur.

3.2.1.1. Variable ordering

The ordering of variables should aim to minimise the potential of splitting the dimension between conceptually similar values. This allows for the creation of more cohesive clusters and limits the potential loss of information.

This paper proposes that the created Visual Generalisation Hierarchies are traversed using Postorder Traversal. In Postorder Traversal [16] a tree is first recursively traversed in the left subtree and then the right subtree. Finally, the parent node is observed. While tree traversing is in terminology most associated with binary trees conceptually, it can be used with non-binary trees, which the created VGHs are.

An ordered list is created during the traversal of the VGH, such that each node's value, which is also a value for the attribute in the dataset, is added to the list. This ordering ensures that similar concepts are, in most cases, placed close to each other in the ordered list. Later, this list can be used to order all the occurrences in the input data.

In Figure 10, an example Value Generalisation Hierarchy for operating systems is visualised in order to demonstrate how such a VGH could be traversed.

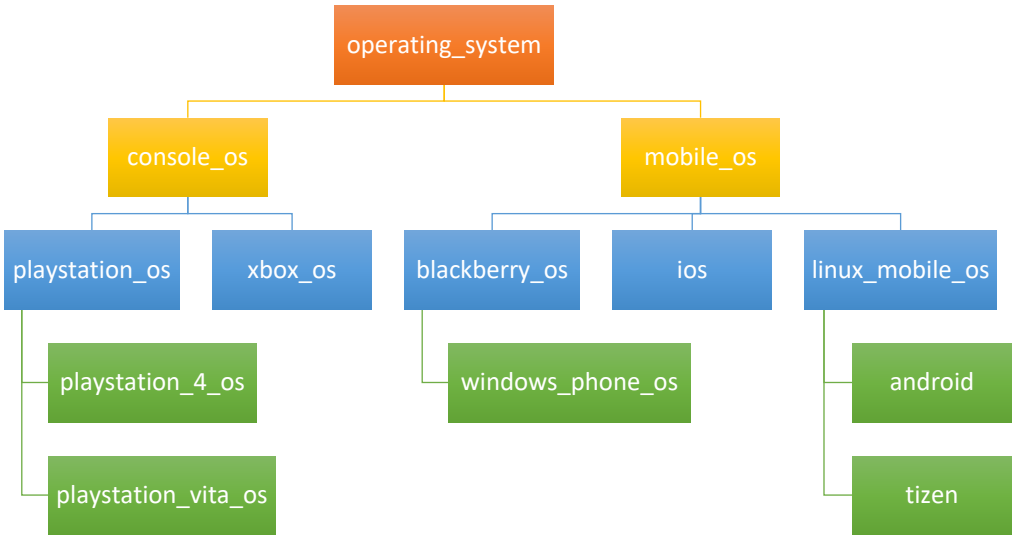


Figure 9. Example Value Generalisation Hierarchy for operating systems

In Postorder Traversal [*Ibid.*], as stated, the child nodes of a subtree are traversed first. If all these values presented in the VGH would also exist as values in the dataset, the ordered list created would be the following: [*playstation_vita_os, playstation_4_os, playstation_os, xbox_os, console_os, windows_phone_os, tizen, android, blackberry_os, ios, linux_mobile_os, mobile_ios, operating_system*]. As observed, most values in the ordered list are close to the same values as in the VGH, however, there are some values that are conceptually quite different, yet are next to each other in the ordered list. This issue is tackled in the second proposal in chapter 3.2.2 of the thesis. Another algorithmically complicated approach would be to mix different means of traversal across the subtrees, such that one subtree might be traversed in post-order, and another in pre-order.

3.2.1.2. Splitting dimensions

In the Mondrian algorithm, it is necessary to evaluate which dimension to split during each recursion. In this proposal, the dimension chosen for splitting is the one that maximises the average distance to the centroid of the dimension.

The point to split the dimension is chosen as the centroid value (see chapter 2.2.3.2 for calculating the centroid). Probabilistically, the centroid value will regularly coincide with the mode of the dimension; however, not necessarily. Thus, often similar centroids as in the *k*-modes algorithm [14] are chosen. If the centroid does not exist within the ordered values

(for example, it is a distant LCS of the values in the cluster), then the closest existing value to the centroid is used as the split value. In an ordered list, an attempt is made to split the dimension before the first occurrence of the value; if either the left or right partition contains less than *k* values, then a cut is attempted after the last occurrence of the value. If neither cut is possible, the next dimension is chosen.

Thus, a dimension of values [*'wolf', 'dog', 'dog', 'dog', 'cat', 'cat'*] and *k* equal to 2, with centroid *dog*, is first attempted to be split into [*'wolf'*] and [*'dog', 'dog', 'dog', 'cat', 'cat'*]. However, as the created partitions do not satisfy the *k* condition, another cut is performed instead to divide the dimension as [*'wolf', 'dog', 'dog', 'dog'*] and [*'cat', 'cat'*].

3.2.2. Semantic Adaptive Multidimensional Strict Mondrian adaptation proposal

The second proposed algorithm in the thesis is the Semantic Adaptive Multidimensional Strict Mondrian algorithm, which is inspired by the SA-MDAV algorithm [6] and shares the same structural basis as the previously discussed Mondrian algorithm (see 3.2.1).

As previously discussed, the strict Mondrian algorithm is already adaptive in the way that is discussed in the SA-MDAV proposal [*Ibid.*], as it disallows cuts within a dimension that splits equal values into different partitions [7]. However, the algorithm's performance can be increased if we pre-process the data in the same way as Sánchez et al. [6], wherein the input data is categorised by their frequencies beforehand. In this regard, the dimensions under observation are much smaller (containing less data points), which allows different operations to be performed faster on the data. Furthermore, it lays the groundwork for the split measure introduced in this thesis.

3.2.2.1. Splitting dimensions

The logic for choosing the dimension to split is the same as the one described in chapter 3.2.1.2. of the thesis. However, this chapter introduces a new proposal for splitting the dimension as an addition to the algorithm.

Ordering data is an integral part of the Mondrian algorithm and ordered lists of values are used in previous works regarding both numerical [7] and mixed or categorical data [15], as well as in the proposal in chapter 3.2.1. In this chapter, an addition is proposed to the algorithm, which lessens the importance of the ordered list; however, ordering the list prior

with some logic still produces better results. In this thesis, the dimension is still sorted by the same measures proposed in chapter 3.2.1.1.

In this proposed addition, rather than relying on the mode or centroid for determining where to split the value, the chosen dimension is iterated over before splitting it to find the index where two consecutive values are furthest apart. The split logic is the main improvement of the algorithm, as it ensures that in most cases, most semantically distant values are separated into different clusters. Thus, emphasis is put on the semantics of the data when choosing the split value and creating the clusters.

Wu Palmer distance, described in chapter 2.1.3 of the thesis, was used as the distance measure. Below is a demonstrative piece of pseudocode, which illustrates how a dimension can be evaluated, where the *sort()* function sorts the dimension based on the post-order traversal of the attribute in the Value Generalisation Hierarchy, and *WuPalmer()* finds the Wu Palmer distance of two values.

ALGORITHM 4. DIMENSION EVALUATION

```
    Input: D (dimension), k (level of anonymity)
    Output: leftPartition, rightPartition
1   sort(D)
2   set splitIndex; foundDistance as 0
3   for (value in D) do
4       set tempDistance as WuPalmer(value, nextValue)
5       if (tempDistance < foundDistance) then
6           set foundDistance as tempDistance
7           set splitIndex as indexOf(value)
8       endif
9   endfor
10  set leftPartition as D[0: splitIndex + 1]
11  set rightPartition as D[splitIndex + 1:]
12  return leftPartition, rightPartition
```

3.3. Developed code

As part of this thesis, multiple microaggregation algorithms were implemented using the Python programming language. The source code is added as Appendix B. In this chapter, an overview of the produced code is given, as well as some concrete examples.

3.3.1. Algorithms

The main algorithms programmed were SA-MDAV (see 2.2.3.), Multidimensional Strict Mondrian (or later, simple Mondrian) and SA-Mondrian (see 3.1.).

Further, as control algorithms, S-MDAV and S-MDAV-Static can be found in the source code and part of the presentation of the results. For testing measures, Mondrian and MDAV for numerical data were also implemented but not used in the thesis for any data gathering, as well as a relaxed Multidimensional Mondrian algorithm.

3.3.2. Specific functions

In this paragraph, two functions are shown, as they are integral to gathering results in this thesis. First, Figure 10 shows the Python implementation of the Weighted Semantic Distance measure described in chapter 2.2.3.1.

```
def _get_weighted_semantic_distance(self, a, b, a_freq, b_freq, tree):
    lcs_depth = tree.depth(self._get_lcs(a, b, tree).identifier) + 1

    a_depth = tree.depth(a.identifier) + 1
    b_depth = tree.depth(b.identifier) + 1

    similarity = 2 * lcs_depth / (a_depth + b_depth)
    weighted_semantic_distance = a_freq * b_freq * (1 - similarity)

    return weighted_semantic_distance
```

Figure 10. Python implementation of Weighted Semantic Distance

In the function, the depth of the compared values is found using the the tree. In addition, the depth of the Least Common Subsumer is calculated with the help of another function. That function is visualised in Python code on Figure 11.

```python
@classmethod
def _get_lcs(cls, node_one, node_two, tree):
    if node_one.is_root():
        return node_one
    elif node_two.is_root():
        return node_two

    set_one = set([node_one])
    set_two = set([node_two])

    while True:
        if node_one in set_two:
            return node_one
        if node_two in set_one:
            return node_two

        node_one_parent = tree.parent(node_one.identifier)
        node_two_parent = tree.parent(node_two.identifier)
        if node_one_parent is not None:
            node_one = node_one_parent
        if node_two_parent is not None:
            node_two = node_two_parent

        set_one.add(node_one)
        set_two.add(node_two)
```

Figure 11. Python implementation of finding the Least Common Subsumer

3.3.3. Assisting code

In addition to the algorithms and functions that were used in them, various other code had to be written in order to achieve the goals of the thesis.

More specifically, functioning code was required, for example, to read, write and modify CSV data, to create Value Generalisation Hierarchies and to plot and measure the results.

The code for the algorithms themselves stays relatively true to the pseudocodes proposed throughout the thesis.

4. Results and evaluation

In this chapter, the results of the thesis are presented and discussed. More specifically, in chapter 4.1. the utility and privacy of the produced anonymised datasets is assessed in terms of information loss and global risk. In chapter 4.2. the algorithm execution times are visualised and reasoned.

For easier readability, the Mondrian algorithm proposed in 3.2.1. is referred to as the simple Mondrian, and the Mondrian algorithm proposed in 3.2.2. is referred to as the semantic adaptive Mondrian in the following parts of the thesis. Note that in the graphs, they are marked as Mondrian-Strict-Improved and SA-Mondrian-Strict respectively.

The results presented in this paragraph were gathered using the different datasets described in Chapter 3.1. of the thesis.

4.1. Utility and privacy

In 4.1.1. graphs related to information loss are visualised, and in 4.1.2. a simple comparison of global risk between the different algorithms is presented.

4.1.1. Information loss

In this chapter, the results regarding information loss are presented. The loss calculation measure used in this thesis is described in 2.2.3.3.

The information loss was calculated for each dataset (*Adult Census*, *Analytics, Analytics Sample One,* and *Analytics Sample Two*). The following four graphs (Figure 13 and Figure 14) show the loss for *k* values ranging from 2 to 10.

Result graphs

Figure 13 shows data loss in *Analytics Sample One* and *Analytics Sample Two,* respectively. In addition to the SA-MDAV algorithm and the two proposals in this thesis (Mondrian-Strict-Improved and SA-Mondrian-Strict on the graphs), multiple other algorithms were executed as a control. S-MDAV and SA-MDAV-Static both originate from the same proposal as SA-MDAV [6], wherein the S-MDAV algorithm does not ensure that all equal values are added into the same cluster, and the SA-MDAV-Static algorithm does not recalculate the centroid on each cluster modification. The visualised Mondrian-Strict algorithm is theoretically the same as the Mondrian-Strict-Improved algorithm. However,

the categorical data is not ordered by any logic; thus, the splits cannot ensure cluster cohesion; rather, the goal is just to satisfy *k*-anonymity.

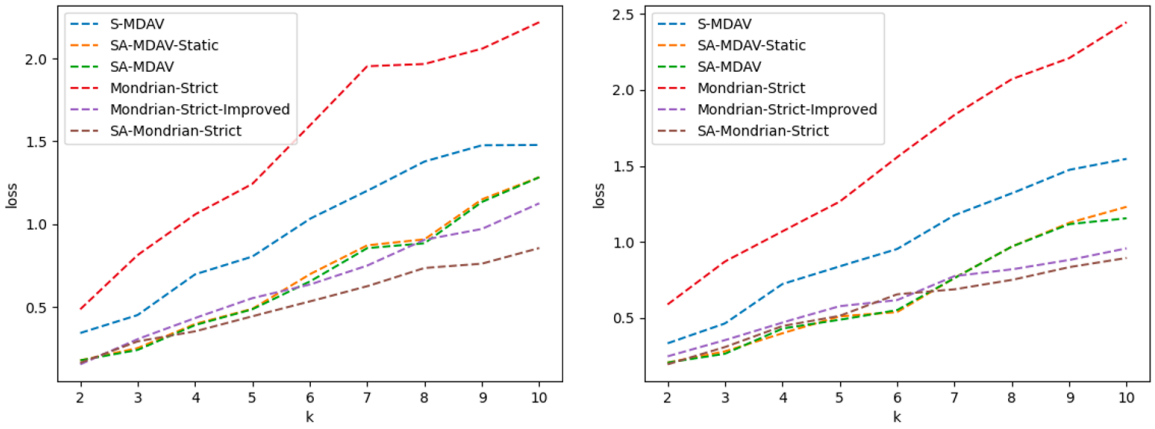


Figure 12. Data loss in relation to k values for Analytics Sample One and Two

Based on the information in Figure 13, our controls function as expected. We can visually deduce that the ordering logic described in 3.3.2.1 significantly increases the output data utility compared to a zero-knowledge random ordering. Across all *k* values, the information loss was less than half in comparison. For the other MDAV algorithms, it is concluded they perform similarly, as demonstrated by Sánchez et al. [6].

While on most occasions, the semantic adaptive Mondrian algorithm performed better than its counterpart, in Figure 13, we can see that the simple Mondrian algorithm outperformed it for a single *k* value of 6. However, we can observe that, in general, the algorithms perform in a seemingly similar manner. The Mondrian implementations at large both also performed better than the SA-MDAV algorithm. Despite this, for some smaller *k* values, SA-MDAV can perform better. Such as in the case where *k* is equal to 3 in *Analytics Sample One*, and k values 3 to 6 in *Analytics Sample Two*.

As our controls showed expected results, the following two results based on the two more extensive datasets are graphed in conjunction with only the SA-MDAV and the two proposed Mondrian algorithms for easier visual comparison. In the following two graphs (Figure 14), the same *k* values (2 – 10) are used; however, the datasets under observation are the entire *Analytics* dataset and the *Adult Census* dataset.

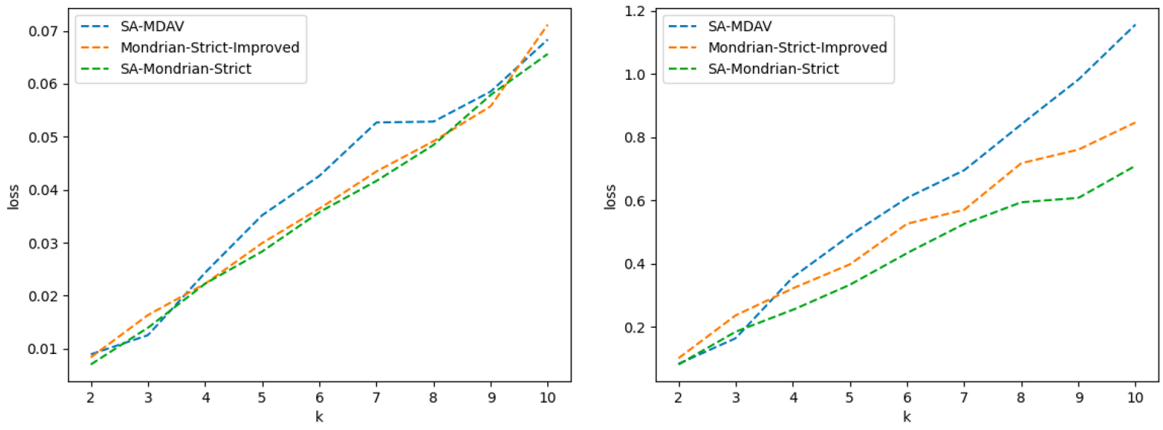


Figure 13. Data loss in relation to k values for Analytics and Adult Census dataset

For most *k* values in both datasets observed in Figure 14, the Mondrian proposals outperform the SA-MDAV algorithm. An interesting observation in the results is that for *k* value of 3 in both datasets, SA-MDAV performs better than the Mondrian algorithms. Further, SA-MDAV outperformed the simple Mondrian on one more case for each of the datasets (*k* equal to 10 in *Analytics* and *k* equal to 2 in *Adult Census* dataset). The worst performance for the semantic adaptive MDAV algorithm was observed in conjunction with *Adult Census* – with *k* equal to 10, the loss when running the semantic adaptive MDAV was almost twice as high as the loss with semantic adaptive Mondrian. However, the information loss values for the *Analytics* dataset remained relatively close for all of the three implemented algorithms.

Both Figures 13 and 14 show that the implemented Mondrian algorithms fare well in comparison to the SA-MDAV, wherein across the used datasets for most *k* values, the semantic adaptive Multidimensional Mondrian algorithm proposed in chapter 3.2.2. outperforms the other algorithms (regarding information loss).

Analysis of results

The results coincide with some previous works; for example, LeFevre et al. [7] also noticed that often the Mondrian algorithm produced better results compared to more expensive recoding models.

During the implementation of the algorithms for this thesis, it was also observed that for SA-MDAV, the higher loss values were often due to a few individual clusters introducing high data loss. If only values with low frequencies are left during the SA-MDAVs cluster creation process, some clusters with high data loss might be created. For example, suppose

there are still enough records to form a cluster. In that case, the records are more likely to be conjoined to form a new cluster rather than being appended to an existing, more giant cluster, even if the values might be semantically distant. The conjunction takes place because the frequency parameter in the weighted semantic distance (see 2.2.3.1) makes it so that the semantics might be overlooked if values with high and low frequencies are compared to each other. The distance is considered bigger due to the weight property of the distance measure.

In the Mondrian counterparts, these values are more likely to end up in more prominent clusters that are more semantically similar. Thus, in the Mondrian algorithms, such data is more often suppressed rather than creating clusters that are generalised to a significant factor. This, in turn, also lessens the potential risk of reidentification, as new value pairs are less likely to be introduced in the data.

In terms of information loss, a potential improvement to the semantic adaptive MDAV algorithm might be to detect when the dataset reaches a state where only values with a certain frequency threshold exist. Then the low-frequency values can be added to the closest clusters at a prior point, as opposed to only once they cannot form a new cluster among themselves. Based on the characteristics of the data, the weighted semantic distance might not be the optimal choice for a distance measure, and it is best served in instances where data is more evenly distributed.

The fact that the simple Mondrian adaptation can, in some cases, outperform the semantic adaptive variation in data loss is explained by the fact that Mondrian is a greedy algorithm. The order of the split dimensions and the precise split points can already be heavily affected by how the previous splits are done. This might create a situation where the first split to make is optimal from the dimension's perspective but not optimal when it comes to clustering the entire dataset. However, the results show that the more advanced approach still results in better outcomes in most cases.

4.1.2. Global risk

In this chapter, the global risk and information loss both are shown in a table setting for each of the three algorithms. The output was produced for the *Adult Census* and the *Analytics Sample One* dataset using k values 2, 6 and 10. The global risk was calculated using RStudio's sdcMicro package.

***Adult Census* dataset**

k	*Global risk*	*Loss*	*Algorithm*
-	1.31 %	0.000%	Before microaggregation
2	1.08%	0.082%	SA-Mondrian-Strict
6	0.66%	0.434%	SA-Mondrian-Strict
10	0.48%	0.710%	SA-Mondrian-Strict
2	1.08%	0.101%	Mondrian-Strict-Improved
6	0.65%	0.526%	Mondrian-Strict-Improved
10	0.46%	0.847%	Mondrian-Strict-Improved
2	1.16%	0.084%	SA-MDAV
6	0.78%	0.608%	SA-MDAV
10	0.60%	1.116%	SA-MDAV

Table 5. Global risk recordings for the Adult Census dataset

***Analytics Sample One* dataset**

k	*Global risk*	*Loss*	*Algorithm*
-	3.80%	0.000%	Before microaggregation
2	1.08%	0.162%	SA-Mondrian-Strict
6	0.66%	0.533%	SA-Mondrian-Strict
10	0.48%	0.856%	SA-Mondrian-Strict
2	1.08%	0.154%	Mondrian-Strict-Improved
6	0.65%	0.635%	Mondrian-Strict-Improved
10	0.46%	1.112%	Mondrian-Strict-Improved
2	1.16%	0.178%	SA-MDAV
6	0.78%	0.653%	SA-MDAV
10	0.60%	1.130%	SA-MDAV

Table 6. Global risk recordings for the Analytics Sample One dataset

In Tables 5 and 6, we can observe the expected results in the sense that with higher *k* values, the risk of reidentification is also lower. While the risk and *k* ratio are similar for the Mondrian adaptations, it is important to notice the relatively higher global risk for the SA-MDAV algorithm. This further implies and confirms the assessment in chapter 4.1.1 that for the Mondrian algorithms, more suppressions rather than generalisations take place. As often, these generalisations can bring new values to the produced dataset that did not exist in the original data, which in a mathematical sense, increases the risk of reidentification as more unique values are present. However, these results might also differ in the case of more evenly distributed datasets.

4.2. Algorithm execution times

The algorithms under observation were all programmed in Python; thus, a general outline of the potential differences between run-times can be assessed. However, the results are still not definite, as some specific implementation choices can significantly affect execution times. Regardless, the measurements can offer some insight.

The same *k* values [2 – 10] are used in presenting the execution time, as were used in chapter 4.1.1. to describe the information loss in the dataset.

First, in Figure 15, the execution times in seconds are visualised for the three algorithms when running on the smaller (*Analytics Sample One* and *Analytics Sample Two)* datasets.

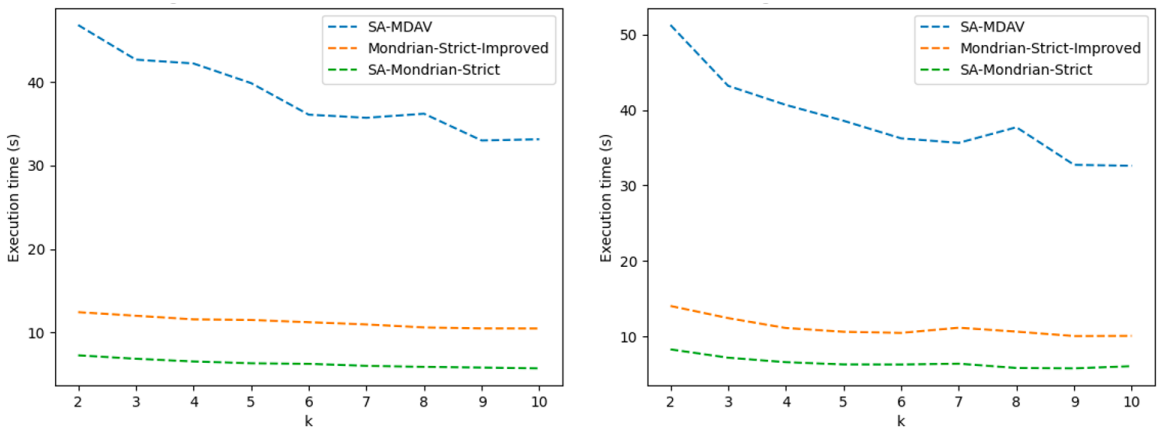


Figure 14. Analytics Sample One and Analytics Sample Two execution times

In Figure 16, the left shows the execution times for the Analytics dataset and the right for the Adult Census dataset.

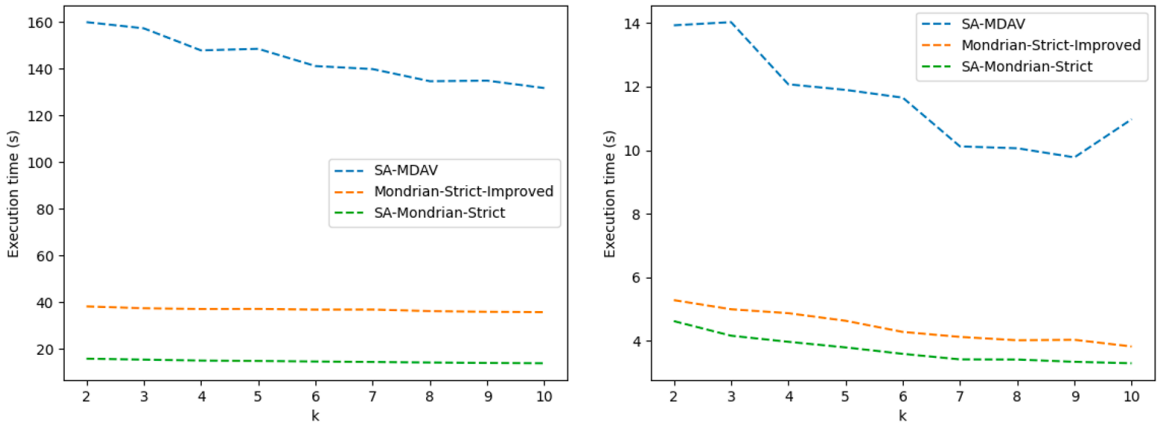


Figure 15. Analytics and Adult Census execution times

For all datasets, we can observe that the execution times are relatively slow for the SA-MDAV algorithm. For the *Analytics* dataset with *k* value 2, the execution time is four times longer than the one for the simple Mondrian algorithm and more than eight times higher than the one for the semantic adaptive Mondrian. This is mainly because the *Analytics* dataset contains many unique values, and the clustering process for the MDAV algorithm is pretty resource intensive.

In SA-MDAV, in each iteration, the distance of a value is calculated concerning all other data. In the case of semantic adaptive Mondrian, the distances are a) only calculated for a single dimension at a time and b) the number of distance measurements equals the number of unique values in the dimension. The semantic adaptive Mondrian algorithm outperforms its Mondrian counterpart because of the pre-processing of data introduced by Sánchez et al. [6], where prior to running the algorithm, values are mapped to their corresponding frequencies. This makes any iteration over the data significantly faster, especially if any values occur with a significant frequency.

The simple Mondrian algorithm could also be easily improved execution time wise by the data pre-processing. It would most likely perform better than the semantic adaptive Mondrian, as no distance measuring during the algorithm execution would be required.

4.2.1. Machine specifications

All results were gathered on the same machine, and in the same condition, with all background applications being closed.

Device: MacBook Pro (16-inch, 2019)
Operating System: macOS Monterey (version 12.4)
Processor: 2,6 GHz 6-Core Intel Core i7
Memory: 16 GB 2667 MHz DDR4
Graphics: AMD Radeon Pro 5300M 4 GB; Intel UHD Graphics 630 1536 MB

The code was executed in the Visual Studio Code (2022) console.

5. Conclusions and future development

The thesis aimed to describe, implement and compare different microaggregation algorithms for categorical data. In addition, as part of the thesis, multiple new proposals for handling categorical attributes with Mondrian were introduced and evaluated. The compared algorithms were implemented and tested in Python. The primary reference point for the algorithms was the semantic adaptive MDAV algorithm. The work on the new proposals proved fruitful. In many cases, the newly proposed algorithms fared well compared to previously proposed microaggregation algorithms regarding information loss, reidentification risk, and execution time. The goals set in this thesis were achieved.

In future works, more research into microaggregation with mixed data might be made, as current works mainly focus on either numerical or categorical data. Thus, appropriate distance and evaluation measures should be trialled to propose functional algorithms.

In consideration of the current proposals, different distance measures could be tested and proposed along with more persuasive logic for splitting the dimensions in the Mondrian algorithms, which would not give up after one or two failures to split the dimension. More algorithms could be adjusted to use the measures in this work and previous works to compare different microaggregation algorithms further.

It is also essential to discuss how to better handle (or incorporate in a mixed manner) categorical variables which are limited to only a few values, such as gender. In these cases, a Value Generalisation Hierarchy cannot be used. In addition, work on improving the semantic Adaptive MDAV algorithm could also be done, such as avoiding grouping low-frequency data points together (and generalising them) and rather suppressing the values.

Bibliography

[1] Domingo-Ferrer, J., Torra, V. Ordinal, Continuous and Heterogeneous *k*-Anonymity Through Microaggregation. *Data Min Knowl Disc* **11,** 195–212 (2005). https://doi.org/10.1007/s10618-005-0007-5

[2] Samarati, Pierangela and Latanya Sweeney. "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression." (1998).

[3] Domingo-Ferrer J. Microaggregation. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA (2009. https://doi.org/10.1007/978-0-387-39940-9_1496

[4] Princeton University "About WordNet." WordNet. Princeton University. 2010.

[5] Wu, Zhibiao & Palmer, Martha. Verbs Semantics and Lexical Selection. Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics. 133-138 (1994). 10.3115/981732.981751.

[6] Martínez, Sergio & Sánchez, David & Valls, Aida. (2012). Semantic Adaptive Microaggregation of Categorical Microdata. Computers & Security. 31. 653-672. 10.1016/j.cose.2012.04.003.

[7] Kristen LeFevre, David J. DeWitt, and Ragu Ramakrishnan. (2006). Mondrian Multidimensional K-Anonymity. In Proceedings of the 22nd International Coference on Data Engineering (ICDE '06). IEEE Computer Society, USA, 25. https://doi.org/10.1109/ICDE.2006.101

[8] Abril, D., Navarro-Arribas, G., Torra, V. (2010). Towards Semantic Microaggregation of Categorical Data for Confidential Documents. In: Torra, V., Narukawa, Y., Daumas, M. (eds) Modeling Decisions for Artificial Intelligence. MDAI 2010. Lecture Notes in Computer Science(), vol 6408. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-16292-3_26

[9] Teorey, T., Lightsone, S., Nadeau, T., & Jagadish, H. V. (2011). *Database Modeling and Design (Fifth Edition).* Burlington, Morgan Kaufmann. 13-34. https://doi.org/10.1016/B978-0-12-382020-4.00002-1

[10] Brambilla, M., and P. Fraternali. (2015). Interaction Flow Modeling Language: Model-Driven UI Engineering Of Web And Mobile Apps With IFML. Burlington, Morgan Kaufmann. 25-50. https://doi.org/10.1016/B978-0-12-800108-0.00003-5

[11] Ayala-Rivera, V., McDonagh, P., Cerqueus, T., & Murphy, L. (2014). A Systematic Comparison and Evaluation of k-Anonymization Algorithms for Practitioners. *Trans. Data Priv., 7*, 337-370.

[12] Hettich S, Bay SD. (1999). The UCI KDD archive, http://kdd.ics.uci.edu.

[13] Rsquared Academy. Handling categorical data in R. (2021) https://github.com/rsquaredacademy-education/online-courses/commit/c777df177b3a6bc2845418c05270c0c5784f7d16.

[14] Torra, V. (2004). Microaggregation for Categorical Variables: A Median Based Approach. In: Domingo-Ferrer, J., Torra, V. (eds) Privacy in Statistical Databases. PSD 2004. Lecture Notes in Computer Science, vol 3050. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-25955-8_13

[15] Böhler,, J. (2021). Multi-Owner data Sharing for Analytics and Integration respecting Confidentiality and OWNer control. D 5 . 5 Report on Data Sanitisation and Computation. MOSAICrOWN. Ref. Ares(2021)6716240 - 31/10/2021.

[16] Mehta, Dinesh P. (2018), "Trees" , in *Handbook of Data Structures and Applications* ed. Dinesh P. Mehta and Sartaj Sahni (Boca Raton: CRC Press, 07 Mar 2018), accessed 10 July 2022 , Routledge Handbooks Online.

Appendices

A. Datasets

This thesis includes a .zip file (*all_files.zip*) with all the datasets used in the creation of the thesis. The datasets are in three different folders.

- **data_files/raw_data:** contains the raw datasets, prior to any modifications
- **data_files/modified_data:** contains the modified datasets for obtaining the results
- **data_files/test_data:** contains datasets that were used during the testing phases of the thesis
- **data_files/output_data:** contains datasets produced to measure global risk in RStudio

All the datasets are Comma-Separated Values (CSV) files.

B. Source code

This thesis includes a .zip file (*all_files.zip*) that includes the Python source code files with implementations of different algorithms discussed in the thesis. Along with different helper files that assisted in gathering the results.

- **src/plot.py:** contains the code to execute and plot SA-MDAV and the two Mondrian proposals for *k* values of 2 to 10, the default dataset used is *Adult Census*
- **src/sa_mdav.py:** contains the semantic adaptive MDAV implementation
- **src/sa_mondrian.py:** contains the semantic adaptive Mondrian implementation
- **src/mondrian.py:** contains the simple Mondrian implementation
- **src/microaggregate.py**: contains some common code used for multiple algorithms
- **src/generalisation_trees.py**: contains the code to create the VGH's
- **src/order_mapping.py**: contains some potential order mappings for the Mondrian implementations (hard-coded)
- **src/create_sample.py**: contains code to create a 10000-record sample of the *Analytics* dataset
- **src/speed_comparison.py**: contains code to execute and plot the SA-MDAV and the two Mondrian proposals in terms of execution time for *k* values of 2 to 10, the default dataset is *Adult Census*
- **src/clean_adult_dataset.py:** contains code to transform the original *Adult Census* to the one used in this thesis (transformations and removal of n/a values)
- **src/clean_analytics_dataset.py:** contains code to transform the *Analytics* dataset to the one used in this thesis (transformations and removal of unused attributes)
- **src/graveyard:** contains some developed, but unused code for the final results, along with the implementations of S-MDAV, SA-MDAV-Static and Mondrian and MDAV for numerical data, along with multiple testing files

Python **3.9** was used to implement the algorithms, to run the code, it is required that the used dependencies are installed. The dependencies are mapped in the *requirements.txt* file and can be installed in a Python environment variable either with the use of *pip* or *conda*.

Glossary

MDAV (algorithm) – Maximum Distance to Average Vector

CSV (file) – Comma-separated Values

VGH – Value Generalisation Hierarchy

SSE – Sum of Squares Error

SST – Sum of Squares Total

SDC – Statistical Disclosure Control

LCS – Least Common Subsumer

WSD – Weighted Semantic Distance