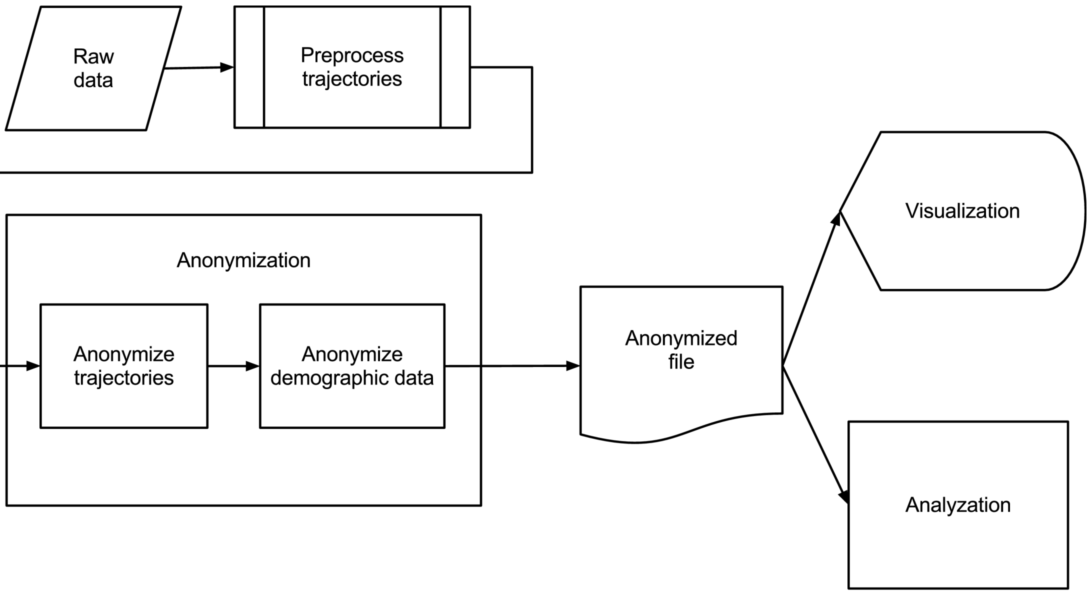
CHALMERS



Methods for Anonymizing Patterns of Human Mobility

Master of Science Thesis in Complex Adaptive Systems

MARTIN NORDSTRÖM

Department of Energy and Environment
Division of Physical Resource Theory
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2012

Report No. 2012:2

Report No. 2012:2

Methods for Anonymizing Patterns of Human Mobility

MARTIN NORDSTRÖM

Department of Energy and Environment
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Methods for Anonymizing Patterns of Human Mobility
MARTIN NORDSTRÖM

Technical report no 2012:2
Department of Energy and Environment
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover: An overview of how the full anonymization presented in this thesis is performed.

Printed by Chalmers Reproservice
Göteborg, Sweden 2012

Methods for Anonymizing Patterns of Human Mobility
MARTIN NORDSTRÖM
Department of Energy and Environment
Division of Physical Resource Theory
Chalmers University of Technology

Abstract

In recent years, the ability to efficiently gather location information of individuals has gained a lot of attention in the research community. There are multiple methods for collecting this data, but this thesis primarily considers data collected from base stations connected to the mobile phones used by people today. Because many users use mobile subscriptions, the demographic data of the users can be collected as well.

However, to maintain the privacy of the individual, the collected data must be anonymized. The aim of this master's thesis is to develop a method to anonymize the data so that it is not possible to identify an individual with a probability above a certain threshold, while still preserving as much information as possible.

The anonymization is mainly divided into two parts. The first part anonymizes the data containing the movement of individuals, while the second part anonymizes the demographic data. The principle of k-anonymization was applied in both parts, which means that each entry in the output of the anonymization is indistinguishable from $k-1$ other entries. Hence, it is only possible to identify an individual with a probability of at most $1/k$. For the anonymization of the demographic data a genetic algorithm was used which minimizes a new definition of information loss which is presented in this thesis. This definition was derived using the Kullback information.

Keywords: Inference attacks, information theory, k-anonymity, location information, location privacy, spatial databases

Acknowledgements

I would like to express my gratitude to my supervisor, Olof Görnerup, for his valuable guidance and suggestions during the entire project. I would also like to thank *Swedish Institute of Computer Science* for giving me the opportunity to do my research work.

Contents

1 Introduction

Today it is possible to collect and share data of where people are and how they move in general. The demographic data of each individual, such as age, gender and income, can be collected as well. There are several use cases for analyzing this data, e.g. to find out which sequences people often travel in order to optimize public transport or where it is most profitable to place specific advertisement banners aimed at people with specific demographic data.

However, one must not forget to preserve the personal integrity. Therefore the collected data of where people are must be anonymized before it is shared to third-parties. Even though names and other data which can be directly linked to an ID (such as the home address) are removed from the anonymized set (the output of an anonymization algorithm), there are still other ways to figure out which individual is associated to some specific data, if it is not anonymized correctly. If one succeeds in obtaining the ID of someone together with a sensitive data of a location, from an anonymized set, then one has performed a so called *inference attack*. Sensitive data or *sensitive value* is an attribute which must not be linked to a specific person in order to preserve the privacy.

One way of performing an inference attack is to link so called *quasi-identifiers* to a specific individual. A quasi-identifier is information about an individual which cannot, by itself, be linked to a unique ID. However, if several quasi-identifier are used in combination it can be linked to the individual in question.

If you have enough quasi-identifier you can say for certain that there is only one individual who has the properties given by the quasi-identifiers. Hence, if you ever, with high probability, can say that an individual with these quasi-identifiers have traveled a certain route you can strongly suspect that this person was there.

An example of an inference attack through quasi-identifier linking: Say that you have a table containing patients in a hospital with demographic data of each individual, such as birth date and home address, and what disease each patient has. You are given the task of releasing this data for research purposes but you have to respect the personal integrity. Thus the *sensitive value* disease should not be linked to the corresponding individual. If you just remove the name and the home address but the zip code, birth date and disease of each individual is left in the table T, this might be prone to inference attacks. There are often very few people with the same birth date and zip code. Say that an attacker obtains an information table S including name and the corresponding zip code and birth date of people living near the hospital in question. Then the attacker has a high probability of being able to link the quasi-identifiers zip code and birth date in table S with the corresponding attributes in table T to see what individuals name has what disease.

The aim of the master's thesis is to anonymize positional data, with demographic data, such that no inference attacks, of where a specific individual has been, can be performed but, at the same time, preserve as much information as possible that can be used by third-parties. The main approach to achieve this, which is used in this thesis, is to apply k-anonymization, which implies that each entry in the output is indistinguishable from $k-1$ other points. This is described in more detail in Section 3.1.

This master's thesis is part of a project called *Consider8*, which is a collaboration project between *Swedish Institute of Computer Science* (*SICS*) and *Ericsson Research*. In this project, location data from mobile phones is first collected. This data is then mapped to predefined, so called, routes and stations. Sequences of both routes and stations is then created from the collected location data. This data, together with demographic data is then anonymized by the algorithms presented in this thesis. Routes are segments in which the person in question travels along, such as roads. Stations are areas in which many people in general stay in, at least, for a while, for example shopping centers.

The rest of the thesis is organized as follows. First some related work in this area is presented in Section 2. Then the problem and some definitions are formulated in Section 3. After that, all necessary steps required for anonymization of the problem given in this thesis is presented in Section 4. This section also includes some common steps that are used in this area and why some of these does not work for this problem. Two different methods are also described for solving the problem in this section, where each method has its advantages as well as disadvantages. Some experimental evaluation on simulated data is presented in Section 5 and comparisons of the results using the two methods are given. Section 6 includes some discussion of the different methods presented in this thesis as well as suggestions for future work. Lastly, some conclusions of the thesis are given in Section 7.

2 Related work

In recent years, several approaches have been suggested which allow for releases of spatio-temporal data while still preserving privacy.

Several methods for anonymizing entire sequences travelled by individuals have been proposed [1, 7, 16, 11]. One common approach is to anonymize individual sequences which consists of positional data with high accuracy (e.g. GPS coordinates) [1, 7]. In these methods the sequences consists of three-dimensional points, where two of the dimensions represents the position (latitude, longitude) and the third dimension is time. In [1] the anonymization is done such that all unique sequences are clustered together so that there are between k and $2k-1$ nearby sequences in each cluster. These are clustered such that each original position in each cluster is not moved too far. If a sequence has to be moved a total distance that is above a given threshold, in order to be included in the anonymized set, it is completely removed instead. If a total number of removed sequences are above another threshold the algorithm starts all over with a higher value of the first threshold. This method returns an anonymized set where each item is the average trajectory of each cluster, together with the number of trajectories represented by the cluster.

A similar approach is given in [7]. Trajectories are initially clustered, and new randomly generated sequences within the volume contained by the sequences in each cluster are returned. This makes sure that the returning data is anonymized while it still reflects where people have travelled. The number of generated sequences is the same as the number of sequences in the cluster.

In [16] a somewhat different approach is used. They describe a method for anonymizing different sequences consisting of GPS-data mapped to a grid with arbitrarily defined resolution. Each point in the sequence has the time attached to it. The anonymized set consists of a table where each row corresponds to an individual and each column a specific time. The entries in the table are spatial rectangles of where the individual has been located in at the time given by the row and column respectively. The first method in the article describes how this table can be constructed such that k-anonymization is applied for each individual. However, the full k-anonymized table contains sequences which can lead to inference attack and the privacy can be breached by using an attack method presented in the article. So, a second method was developed and presented in the article, which makes sure that the identified attack method does not work and that the anonymized data is safe from these kinds of inference attacks.

Instead of anonymizing sequences directly, one approach is to first define some indivisible parts which is often travelled by people and then map the sequences on these parts. Each part can, for example, be a road segment. This makes it easier for someone who analyzes the anonymized data to know what the sequence refer to and the anonymized data will be consistent for the same travelled road sequence for multiple anonymizations during different time intervals. By using the method described in [11] it is possible to anonymize data which has been preprocessed into these indivisible parts. However, this method can be used for any type of sequential discrete data and is not restricted to the indivisible parts described here. By using this method a tree is usually created, where each branch represents a sequential item. All branches which reaches k-anonymity is then moved to an anonymized tree. Some of the sequences of branches that do not reach k-anonymity are modified to fit into a sequence of the k-anonymized tree.

However, as far as I know, no attempt has been done before which anonymizes sequences, where demographic data of each traveling individual is also known. A method for solving this problem is the foremost contribution of this thesis.

When anonymizing non-sequential data (also known as micro data), k-anonymization is often performed on *quasi-identifiers* after ordinary identifiers have been removed. This approach was first introduced in [13]. Different definitions of the "quality" of a k-anonymization have been proposed before [4, 2, 5, 6] and it has been proven to be NP-hard to solve the minimality of k-anonymization [8].

An incremental dataset is when data is continuously added to some dataset. In order to be able to release multiple releases of the data over time, one has to consider previously released data. In [3] incremental datasets are considered for anonymization. When new data is added to the original (non-anonymized) dataset, it is first placed in a "waiting list" which requires that at least k items are in an equivalence class before it can be released in the anonymized set. They also describes how to split an equivalence class to reach a higher resolution of the released anonymized set while preventing inference attacks. The article only shows examples of how to do this with so called l-diversity (see [3]), but, according to the authors, k-anonymization should be equivalent. However, this article describes datasets which are only incremental and does not describe how to deal with data where some items in the original dataset are removed over time as well.

Multiple releases of the same dataset can be useful if different attributes are presented in different releases. However, due to overlaps between attributes in different releases it might be possible to perform inference attacks. In [14] the (X, Y)-anonymity is described, which is a generalization of k-anonymity. By using (X, Y)-anonymity it is possible to prevent inference attacks which uses different kinds of overlaps between different releases.

In [4] the *classification metric* was introduced for the first time. They split d-dimensional data points (where d is an arbitrary positive integer) in so called *single dimensional partitionings* and use a straightforward genetic algorithm to minimize the classification metric with k-anonymization as constraint. In [2] the *discernability metric* was introduced. In this article, the same type of data is anonymized and single dimensional splits were used as well. Even though the algorithm finds the optimal solution (which is NP-hard), it manages to do so in a reasonable time on realistic data by taking advantage of some properties of the discernability metric. They were also able to do the same for the classification metric. In [6] a greedy algorithm for optimizing the discernability metric is explained. This algorithm produces so called *strict multidimensional partitionings*, and the result from this algorithm is not guaranteed to be optimal for this partitioning. However, because the algorithm produces strict multidimensional partitionings they were able to, in an experimental evaluation, obtain a result which discernability metric value is more optimal compared to the result given in [2] (which returns the *optimal* solution using only the single dimensional partitioning).

3 Definitions and problem formulation

3.1 k-anonymization

k-anonymization is used for the methods in this thesis in order to prevent inference attacks.

Definitions:

- A *Quasi-Identifier* is an attribute of an individual which cannot by itself be used to identify the individual. However, if several quasi-identifiers are joined together it might be possible to identify an individual. This is because a *combination* of a certain set of different quasi-identifiers is represented by only a few individuals. Quasi-identifiers can for example be age, gender and income etc.
- A *Sensitive Value* is the kind of attribute that must not be linked to the corresponding individual, in order to preserve the privacy. If a sensitive value is linked to an individual, an *inference attack* has been performed.
- An *Equivalence class* consists of different intervals in different dimensions, one for each quasi-identifier. If a point is contained by all intervals of an equivalence class, the point is contained by that equivalence class. Thus, an equivalence class is represented by its intervals and the number of points in the interval. For example: 23 individuals with an age of 20–25 years and an income of 3,000–4,000$ per month. In Section 4.4.3, where a new definition of information loss is introduced, the correspondence to the *equivalence class* is called an *anonymized interval*.
- *k-anonymization* clusters the data such that each cluster contains at least k entries. A cluster can for example be an equivalence class.

From these definitions we see that, for each element in the anonymized set, there are at least $k-1$ elements with the same representation (that are in the same cluster). Therefore, each element is indistinguishable from at least $k-1$ other elements. Given this information, the probability to link a specific individual to a *sensitive value* is thus less than or equal to $1/k$. If *suppression* is allowed (ability to remove data) and there are less than k people represented in a cluster, the number of individuals shown in this cluster is truncated to zero after anonymization (or it is not represented at all).

3.2 Problem specific definitions

- *Demographic data* is a set of attributes of an individual, such as the individuals age, gender and income etc. Every trajectory includes a demographic data set. Other data may also be included in this data, even though it is not really demographic data. This data can include information of the trajectory, such as the average speed of the trajectory and at what time the trajectory started and ended.
- A *route* is an indivisible segment in which a person can travel along, such as a road segment. A *station* is an area in which many people in general stay in, at least, for a while, for example shopping centers. Routes and stations are equivalent for the anonymization in this thesis and are therefore always referred to by *routes*.
- A *sequence* is an ordered list of subsequent routes and stations which an individual can travel.
- A *trajectory* is a sequence that a specific person has traveled which contains demographic data. That is, it is a sequence with associated demographic data. It can be either the full sequence that the individual has traveled or a part of the full sequence.
- A *trajectory group* is a group of trajectories that share the same sequence. That is, it is a single sequence together with a number of demographic data representing individuals.

3.3 Problem formulation

The data that will be anonymized is trajectories travelled by people. This problem requires that even with external information, it should not be possible to infer more information, which is considered as sensitive data, about a specific individual from the anonymized set. The anonymized set is the anonymized output of a given algorithm. In this case, sensitive data is when you can conclude, with a probability above a certain threshold, that a specific individual has been or has *not* been traveling a specific sequence.

All data that is attached to each sequence, such as demographic data, should be considered as quasi-identifier because all this data can be obtained externally. This includes subsequent location data, that is, parts of the full sequence. This is included because one can find out that a specific individual has been located on a certain route or station from surveillance cameras etc. Location is the only data that is considered as sensitive data one can obtain from the anonymized set. Hence, location data should be considered both as quasi-identifier and sensitive value. Attributes which has this kind of property has some interesting properties which will be discussed more later on.

A simple example on input trajectories and anonymized output trajectory groups can be seen in Figure 1

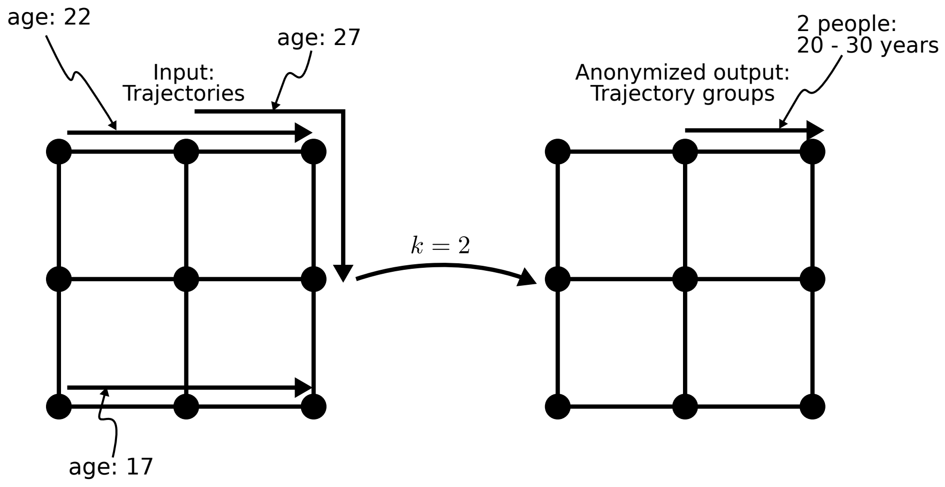


Figure 1: This is a simple example of anonymization of trajectories with demographic data. The circles in this figure represents nodes and the arcs between nodes are routes. The arrows in the left part are original trajectories. The arrows on the right part represents anonymized trajectory groups. The sequence for a trajectory group is given by an arrow. $k = 2$ in this example and because there is only overlap between two trajectories in one route, this is the only sequence that is represented in the anonymized set. The demographic data of the two individuals who travelled this sequence were anonymized and represented as well.

4 Methods

The anonymization consists mainly of two different parts. The first part deals with anonymizing the sequences while the second part takes care of anonymizing the demographic data and makes sure that the output of the entire anonymization procedure is anonymized. The input to the anonymization method is first sent to the trajectory anonymizer. The output of the trajectory anonymizer is the input to the demographic data anonymizer. After that, one to two simple filters are applied to the output of the demographic data anonymizer and the resulting output is the fully anonymized data, see Figure 2. This figure contains, generally, all steps from the collection of data to the anonymized data. The preprocessing of trajectories maps the raw data from mobile phones to travelled trajectories on predefined routes. In this part there is some information loss, so this can also be considered as part of the anonymization. However, the box marked “Anonymization” in this figure shows which parts, of all required steps, that have anonymization as their main purpose. The output of all steps shows XML as an example, but any type of output is of course possible. The output file can then be used for visualization or other types of analyzation methods.

In this thesis, two different methods are presented for the trajectory anonymizer and one method is presented for the demographic data anonymizer. The two methods for the trajectory anonymizer both have their advantages as well as disadvantages compared to each other. However, as long as the prerequisites (described below) of each part is preserved, according to how you define some certain aspects, you can use almost any method for each part and the resulting set should be protected

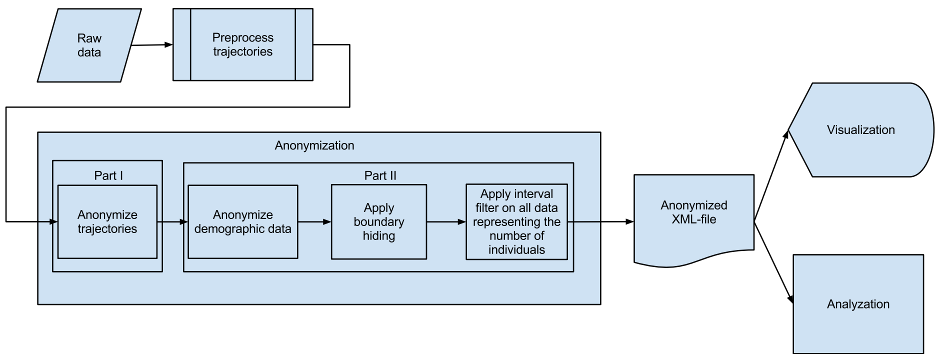


Figure 2: An overview of how the full anonymization is performed. “Anonymize trajectories” and “Anonymize demographic data” are the, so called, main anonymization methods. The interval filter does not have to be applied in some cases.

from inference attacks. This is not proven but I will strongly argue for this claim. In Section 4.3 the two methods of the trajectory anonymization are compared to each other.

Prerequisites for part I, trajectory anonymization:

1. **Input:** A set of trajectories.
2. **Output:** A set of trajectory groups, where each trajectory group is represented by at least k individuals. The sequences in the trajectory groups may be altered in order to fulfill k-anonymization. Trajectory groups which does not reach k-anonymization are suppressed.

All groups in the output of part I are anonymized independently of each other in part II. Hence, part II can be applied to all trajectory groups concurrently.

Prerequisites for part II, demographic data anonymizer:

1. **Input:** Demographic data points belonging to *one* of the groups in the output of part I.
2. **Output:** Equivalence classes with k-anonymized demographic data which also fulfills the criteria that one cannot infer that a specific individual belongs to or does *not* belong to a certain group. This should hold even when combining the output of the demographic data anonymizer from other groups.

These **prerequisites** are very general and more detailed information on how they are used for the task in this thesis is presented below. These are summarized under some **rules**, also given below.

The output after applying the two parts are different trajectory groups containing different sequences, and each trajectory group also contains equivalence classes of demographic data. Each trajectory group represents at least k individuals who travelled the trajectory group’s corresponding sequence (in the given direction), either as a full sequence or as a part of a longer sequence.The demographic data of these individuals are represented by the trajectory group’s equivalence class and the boundaries of the equivalence class’s intervals are placed such that it is not possible to infer an exact data value, see Section 4.4.5.

4.1 Anonymization approach

One approach of preventing inference attacks while still keeping as much information as possible is to let all data go through a filter and, if the filter identifies possible inference attack, the outgoing data is modified such that this kind of inference attack cannot be performed. However, in order to prevent all kinds of inference attacks using this approach, we must first prove that we have found all possible inference attacks that can be performed (which is very often close to an impossible task [13]).

A different approach is to apply the same modification (anonymization) to all data. This modification algorithm is developed such that we can show that the possible inference attacks that we have identified cannot be performed on this modified data. When using this approach, we probably prevent inference attacks which are closely related to the ones we have identified and even others as well. We consider this to be a safer approach compared to identify inference attacks during the anonymization and only modify this data.

In some cases it has been found that it is possible to, in some worst-case scenarios, perform inference attacks on data that have only been anonymized by the two major anonymization methods (trajectory anonymizer and demographic data anonymizer). In these cases we have developed additional filters for anonymization which are to be applied after the main method. Again, even though some of these additional filters are applied to prevent only one single type of inference attack which we have found, all data is modified by these filters. Hence, we *never* try to identify a possible inference attack during the anonymization, but rather rely on that the modification of the data prevents inference attack.

The anonymization is also done such that it considers data of previous anonymized releases. Several approaches take this case into consideration, for example [15, 3]. In these methods, new data is combined with previously released data for a new release and the anonymization directly considers previously anonymized releases such that the new anonymized set is safe from inference attacks. A different possible approach is to use some modification method which is developed such that when applied to some original data during the anonymization, the result is always safe from inference attacks. This should hold regardless if some of the original data has been used in previously anonymized releases or not. We choose the latter anonymization alternative because some inference attacks that would be possible by using data from other sources, together with our anonymized data, is likely to be prevented, see Section 4.5.3.

When applying k-anonymization in all methods described in this thesis, it is required that k different *individuals* were in the trajectory group that will be represented in the anonymized set. However, once this limit has been reached, the number attached to each trajectory group, represents the number of *trajectories* of that trajectory group. Thus, one individual can be represented multiple times in a trajectory group if he or she travelled along that trajectory multiple times. Hence, this preserves the privacy of people who travels unique trajectories many times. For example, if one individual travelled on the same sequence k times where no other people travelled at that time, this will not be shown in the anonymized set. However, if k people travelled on the same sequence once per person and another person travelled on the same sequence k times, this will be shown in the anonymized set as if there were $2k$ trajectories on the corresponding trajectory group. The same holds for demographic data in equivalence classes of a trajectory group. Even though this is sometimes loosely referred to by "k people" or similar, this still holds for all methods in this thesis.

4.2 Interval filter

In order to prevent inference attacks, the filter described here must be applied in some cases. The reason for why it must be implemented is somewhat different for different algorithms and the reason is described whenever the interval filter is required to be applied.

This filter is used in order to cover the *exact* number of *people*. The interval filter simply replaces the number of people in each output of an algorithm with the corresponding interval with size I_s. The intervals, where the first one starts with k, are non-overlapping and are therefore given by

$$[k + I_s i \mathrel{..} k + I_s(i + 1)),\ i \in \mathbb{N}_0 \tag{1}$$

($\mathbb{N}_0 \equiv \mathbb{N} \cup \{0\}$, $[a..b) \equiv \{x \in \mathbb{N}_0 | a \leq x < b\}$).

For example, if $k = 10$ and $I_s = 5$, the first three intervals are represented by the values $\{10, 11, 12, 13, 14\}$, $\{15, 16, 17, 18, 19\}$ and $\{20, 21, 22, 23, 24\}$. If the number of people is 16 and is hidden by the interval filter it is, in this case, replaced by the interval represented by the values $\{15, 16, 17, 18, 19\}$.

4.3 Part I: Trajectory anonymization

Location data is, partially, considered as a quasi-identifier and hence, at least k sequences are required before they can be included (location data is also considered as sensitive value). It is important to keep location data unaltered as this is often the most important data of the input set. If there are less than k instances of the same sequence it cannot be used as a complete sequence but a subsequence may be used instead. These can be added to the anonymized set when there are at least k subtrajectories over the same sequence. We now have the rules for part I.

4.3.1 Rules for trajectory anonymizer

1. The output must be k-anonymized. Therefore the anonymized set contains trajectory groups which each contains k individuals or more.
2. Sequences are here considered as the most important part of the data and are therefore not altered. This means that similar sequences may not be "approximated" to a single sequence and that multiple sequences that has some parts in common may not be concatenated together to a long sequence. The only alteration of sequences that we allow is when multiple sequences share the same subsequence so that this subsequence can be used as the sequence for a trajectory group.
3. A trajectory can be used in multiple trajectory groups if its subsequences are non-overlapping. Overlapping subtrajectories from the same trajectory can only be used if the algorithm prevents inference attacks which exploits overlapping subsequences and only attacks directly on the anonymized set together with publicly available demographic data are considered. Hence, if it is assumed that a possible attacker can get some other external data (such as the number of people who travels on a certain route), overlapping subtrajectories are not allowed.

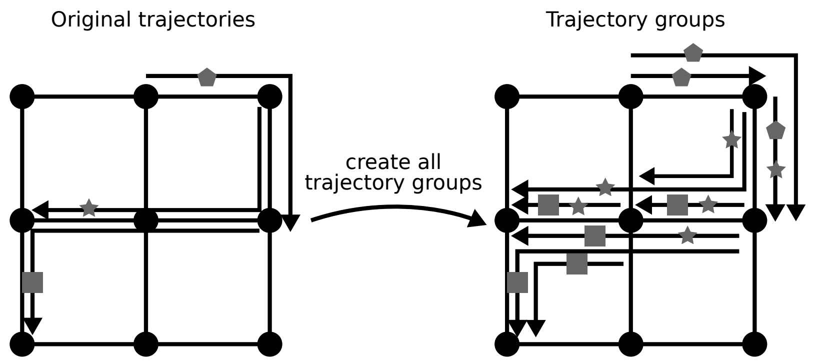


Figure 3: An example of the creation of all possible trajectory groups. The circles in this figure represents nodes and the arcs between nodes are routes. The arrows in the left part are original trajectories, where each individual traveling each trajectory is represented by either a star, a pentagon or a square. The arrows on the right part represents different trajectory groups. The sequence for a trajectory group is given by an arrow, and the shapes (a star, a pentagon and/or a square) on the arrow represents the individuals who travelled along the sequence given by the arrow. k-anonymization has not been applied to the trajectory groups in this figure.

Two algorithms are presented below which both have their advantages as well as disadvantages compared to the other method. In short the *overlapping trajectory anonymizer* is suitable for a high amount of people where it is possible to have a high value of k (order of magnitude at least 10^2) without too much information loss, for example big cities. However, the overlapping trajectory anonymizer has some weaknesses that should be considered before that method is chosen, see Section 4.3.2. The *nonoverlapping trajectory anonymizer* is suitable when the amount of people is moderate and the value of k must be low (order of magnitude 10^1) in order to not loose too much information.

4.3.2 Overlapping trajectory anonymizer

This method has relatively low information loss but some types of inference attacks are possible in some worst-case scenarios. However, for high values of k these types of inference attacks might be considered as tolerable. The method is very simple and works as follows.

The algorithm finds *all* possible trajectory groups where at least k people have travelled. As previously defined, a trajectory is either a part of a sequence travelled by an individual, or the full sequence. Hence, the same travelled sequence can be represented multiple times in multiple trajectory groups. For example, say that $k = 5$ and five people have travelled the exact same sequence, which has a length of ten routes. In that case, when *all* possible trajectory groups are created, the number of trajectory groups is then $(10 + 1)10/2 = 55$.

See Figure 3 for another example where all possible trajectory groups are created. k-anonymization was not applied in this figure. When k-anonymization has been applied, these trajectory groups, together with the people who travelled each subsequence are placed in the anonymized set. Hence, by using this algorithm, all data which corresponds to at least k people have a representation in the output of the algorithm.

This method requires that the *interval filter* (see Section 4.2) is applied in order to prevent inference attacks. The interval size, I_s, is either $k/2$ or k. This depends on what type of inference attacks which are tolerated in some worst-case scenarios. In order to come to this conclusion we must first introduce the demographic data anonymizer. The conclusion of this fact is therefore derived in Section 4.5.1. In this section some other inference attacks are discussed as well and this should thoroughly be considered before making the decision to use this algorithm.

4.3.3 Nonoverlapping trajectory anonymization

This method has higher information loss compared to the *overlapping trajectory anonymizer*, but no inference attack has been found when using this method for anonymization. This holds even for inference attacks that uses information outside the scope of publicly available demographic data connected to a corresponding individual. For example, information from surveillance cameras is considered as information outside this scope.

Because we want as low information loss as possible it is advantageous to put as long and as many trajectories as possible in the anonymized set. To achieve this we do as follows. *All* possible trajectory groups with more than or equal to k individuals are created. This means that all possible trajectory groups with less than k people are not used and are therefore be removed. We call this set of groups the *initial set*. Hence, the initial set of this algorithm is actually the output of *overlapping trajectory anonymizer*.

Each trajectory group is assigned a score. We want to maximize the total score of the anonymized set. Therefore a higher score means that it is more probable for that trajectory group to be included in the anonymized set. The score of the trajectory group can be obtained from any arbitrary function but as was stated earlier we wanted as long and as many trajectories as possible. However, we consider longer sequences to have priority over the number of trajectories of a trajectory group. Therefore we define the following score function:

$$S(G) = n \cdot l^2 \tag{2}$$

where $S(G)$ is the score for the given trajectory group G, n is the number of people in G and l is the length of the sequence which G represents.

This is the score function used when evaluating the algorithm (see Section 5.1).

In order to handle the large number of trajectory groups, we employ a greedy algorithm that if fast, but that does not guarantee to find the optimal solution, given by the maximum total score of all trajectory groups in the anonymized set.

In each iteration of the greedy algorithm, as many trajectories as possible up to k trajectories are extracted from the trajectory group with the highest score and are put in the anonymized set in the corresponding trajectory group. If there has been more trajectories extracted from the same trajectory group earlier, the new extracted trajectories are put in the same trajectory group in the anonymized set. When a subtrajectory t is moved to the anonymized set, all trajectories, which overlaps with t *and* were also created from the same original trajectory as t, are removed. This is done to prevent some inference attacks that uses information from the same trajectory on overlapping subtrajectories which can occur when using the *overlapping trajectory anonymizer*. See Section 4.5.1 for more information about this. This operation can cause some trajectory groups in the initial set to contain less than k trajectories. If this happens and there is no corresponding trajectory group in the anonymized set, all these trajectories are removed immediately (they can never reach k-anonymization). These steps are iterated until there are no trajectories left. Hence, by using this algorithm the resulting anonymized set has the following property: Each trajectory represented in the anonymized set is only represented multiple times if the representations (given by trajectory groups) of the same original trajectory are non-overlapping.

During the execution of the algorithm, if there are more than k trajectories in the trajectory group with the highest score, the trajectories that are moved to the anonymized set are selected at random. The reason for this is that we want the selected trajectories to be unbiased.

There is also a reason for why we choose to only move at most k trajectories at a time, from the initial set to the anonymized set. For example, consider the situation when several long trajectory groups have one route in common. In many cases, the trajectory group with this common route as its sequence, is the top scored trajectory group. If we choose to move *all* trajectories from the top scored trajectory group, then all other trajectory groups passing through this common route (which all have a sequence length of at least two) have to be removed. In that case we lose information of the combination of where people travel before and after this common route. Also, there is a risk that this approach is farther from the optimal solution compared to the approach of only moving k trajectories from the top scored trajectory group at a time. This is the reason for why we choose the latter approach. The optimal solution in this case is the set of trajectory groups in the anonymized set with highest total sum of scores.

The resulting trajectory groups in the anonymized set is the output of this part. Figure 4 shows an example run of this algorithm on a simple example.

Origin – destination groups As stated before, when using this algorithm the information loss is much higher compared to the overlapping trajectory anonymizer. However, it is possible to add additional parts to the nonoverlapping trajectory anonymization algorithm in order to preserve more information while still preserving privacy. One kind of information that is often desired is the origin-destination, that is, where one started a trip and where the trip ended. Because the nonoverlapping trajectory anonymization algorithm only extracts at most some subtrajectories of each full trajectory in the initial set, the information of the entire trajectory is often lost. To keep some information of origin-destination we can add the following part, which is executed after the main part of the nonoverlapping trajectory anonymizer: For each trajectory group in the anonymized set, find all common nearby origin nodes and/or destination nodes. For example, if at least k trajectories in a trajectory group started their sequences in nodes which are all close to each other, this information can be preserved without violating privacy. In principle, any definition of "nodes which are all close to each other" works but an example of a more strict definition is the following: A common group of nodes are one node together with all its neighboring nodes (a neighboring node is connected by a single route).

To avoid overlap, each trajectory may only be included in at most one origin group and one destination group. The information about the common groups are added to the anonymized trajectory group. This in-

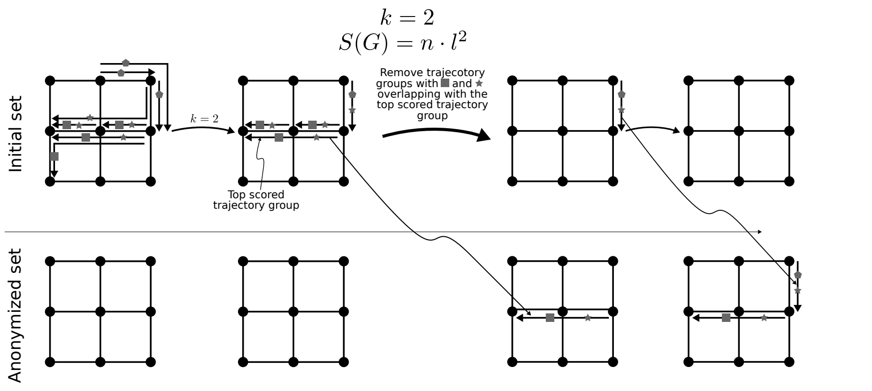

$k = 2$
$S(G) = n \cdot l^2$
Initial set
Anonymized set
$k = 2$
Top scored trajectory group
Remove trajecotory groups with ■ and ★ overlapping with the top scored trajectory group

Figure 4: This is a simple example of the algorithmic steps for the nonoverlapping trajectory anonymizer. The circles in this figure represents nodes and the arcs between nodes are routes. The top left map shows all possible trajectory groups which are created as the first step of the algorithm. The algorithmic steps goes from left to right and the trajectory groups for both the initial and the anonymized set are shown for each step. The anonymized set is initially empty. The first step of the algorithm filters out all trajectory group that cannot reach k-anonymity. In this case it keeps all trajectory groups with at least two represented individuals. After that, all trajectory groups are given a score according to the score function. k individuals from the top scored trajectory group is moved to the anonymized set. All representations of individuals, that are moved to the anonymized set, in trajectory groups that overlaps with the top scored trajectory group, are removed from the initial set. These steps continues until the initial set is empty. The trajectory groups in the anonymized set is the output of the nonoverlapping trajectory anonymizer.

formation contains, for each common group, the nodes in the common group, if it is an origin group or destination group, and how many trajectories in the trajectory group that started or ended their sequence in this common group (which must be at least k trajectories). This is, again, an optimization problem to preserve as much information possible and an approach similar to the main nonoverlapping trajectory anonymization algorithm can be used here as well. For example, give each common group, with at least k trajectories, a score given by some score function and move k trajectories to a new anonymized set from the common group with the highest score.

4.3.4 Algorithms

Pseudo code for both the overlapping trajectory anonymizer and the nonoverlapping trajectory anonymizer are given in algorithm 1 and 4 respectively.

Algorithm 1 Overlapping Trajectory Anonymizer

```
OverlappingTrajectoryAnonymizer[$trajectories$, $k$]
Ensure: $tg.size \geq k \ \forall tg \in tgMap.trajectoryGroups$
  // A hash table where the key is a sequence and the value is a trajectory group:
  $tgMap \leftarrow$ map[$key = Sequence$]($value = Trajectorygroup$)
  // Fill trajectory groups with trajectories:
  for $t \in trajectories$ do
    $subsequences \leftarrow$ getAllSubsequences($t.sequence$)
    $tgMap \leftarrow$ registerTrajectoriesInTrajectoryGroups($t$, $tgMap$, $subsequences$)
  end for
  Remove all trajectory groups in $tgMap$ with less than $k$ individuals
  return $tgMap.trajectoryGroups$
```

Algorithm 2 Returns all possible subsequences

```
getAllSubsequences[$sequence$]
Ensure: $|subsequences| = (n+1)n/2$, where $n = sequence.length$
  $subsequnces \leftarrow \{\}$
  for $j \in 0..sequence.length$ do
    for $i \in 0..j-1$ do
      $subsequences \leftarrow subsequences \cup \{sequence[i:j]\}$
      // $list[i:j]$ is the tuple of items given by the $i$'th item in $list$ to the $(j-1)$'th
      item in $list$
    end for
  end for
  return $subsequences$
```

4.4 Part II: Demographic data anonymization

The output of the first part is multiple trajectory groups. Each one contains individuals with their demographic data. This demographic data contains quasi-identifiers, so each trajectory group must be anonymized. However, there is no direct relation between different trajectory groups. This means that the same demographic data anonymizer can be applied on

Algorithm 3 Registers the given trajectory t in the trajectory groups with given subsequences

registerTrajectoriesInTrajectoryGroups[$t, tgMap, subsequences$]

```
for $subseq \in subsequences$ do
    if $subseq \notin tgMap$ then
        $tgMap[subseq] \leftarrow$ New trajectory group with $subseq$ as sequence
    end if
    // Obtains the value given by $subseq$ as key:
    $tg \leftarrow tgMap[subseq]$
    $tg$.add($t$)
end for
return $tgMap$
```

Algorithm 4 Nonoverlapping Trajectory Anonymizer

NonoverlappingTrajectoryAnonymizer[$trajectories$, k]

```
Ensure: $tg.size \geq k \ \forall tg \in anonymizedTGMap$
    // A hash table where the key is a sequence and the value is a trajectory group:
    $tgMap \leftarrow$ map[$key = Sequence$]($value = Trajectorygroup$)
    // Fill trajectory groups with trajectories and define siblings:
    for $t \in trajectories$ do
        $subsequences \leftarrow$ getAllSubsequences($t.sequence$)
        $subsequences \leftarrow$ registerOverlappingSubsequencesAsSiblings($subsequences$)
        $tgMap \leftarrow$ registerTrajectoriesInTrajectoryGroups($t$, $tgMap$, $subsequences$)
    end for
    $anonymizedTGMap \leftarrow$ map[$key = Sequence$]($value = Trajectory$)
    // Move trajectories from top scored trajectory groups to $anonymizedTGMap$ until
    $tgMap$ is empty:
    while $tgMap \neq \emptyset$ do
        $tg \leftarrow$ Top scored trajectory group in $tgMap$
        $trajectories \leftarrow$ Get up to $k$ trajectories from $tg$
        Remove all siblings to all $trajectories$ from $tgMap$
        Move $trajectories$ from $tgMap$ to $anonymizedTGMap$
        Remove trajectory groups from $tgMap$ which can never reach $k$-anonymization
    end while
    return $anonymizedTGMap.trajectoryGroups$
```

Algorithm 5 All subsequences in the given input which are overlapping have pointers to each other. This is used to ensure that the same trajectory is never used multiple times in overlapping trajectory groups.

```
registerOverlappingSubsequencesAsSiblings[$subsequences$]

  for $subseq \in subsequences$ do
    for $otherSubseq \in subsequences \backslash \{subseq\}$ do
      // if there is some overlap between $subseq$ and $otherSubseq$:
      if $subseq \cap otherSubseq \neq \emptyset$ then
        $subseq.siblings \leftarrow subseq.siblings \cup \{otherSubseq\}$
      end if
    end for
  end for
  return $subsequences$
```

each trajectory group concurrently. There are still some indirect relations between different trajectory groups. How to deal with this is described later on.

The demographic data anonymizer of the algorithm uses k-anonymization to preserve the privacy. Hence, if possible, this method groups points together in intervals containing k to $2k-1$ points (if a group contains $\geq 2k$ points it can, in many cases, be split into at least two groups with $\geq k$ points in each group, therefore obtaining a higher resolution while maintaining k-anonymization). Again, a *point* is the representation of demographic data of an individual. However, in some cases the resolution needed, to split data such that $< 2k$ are in each group, is too high. Therefore it is sometimes not possible to achieve intervals containing between k and $2k-1$ points. Another possibility is that more than $2k$ points have the exact same value, which makes them indistinguishable from start. Therefore, it is not a requirement that each interval contains between k and $2k-1$ points each.

The approach of dividing the points into different equivalence classes is often done such that every equivalence class contains $\geq k$ points, and every point is represented in one equivalence class. Thus, no points are suppressed. In this thesis we will refer to this approach as *interval-anonymization without point suppression*.

Disadvantages for interval-anonymization without point suppression Interval-anonymization without point suppression (where no points are suppressed) has however, several disadvantages, especially in our problem. If a data point has relatively unique data, that is, it is far from most other points for at least one parameter, this point must still be included in an interval. This will result in that the interval of the group that it belongs to is relatively large in at least one dimension (one parameter) because of the data point's relatively unique value. Thus, in some cases, it is better to remove so called *outliers* (points far away from other points) so that the resolution of the groups is higher compared to keeping the outliers in the groups.

Another disadvantage of using interval-anonymization without point suppression is that it often assumes that the attacker already knows that the individual that he wants to attack is in the set. In our case this is not always the case. We don't want an attacker to infer from the anonymized set that a specific individual has traveled a specific sequence where the only prior knowledge is this individual's demographic data. In some cases the demographic data of an individual is relatively unique. For example, say that we have the two-dimensional space (age, income) and that there is an individual with relatively high income at a low age, which in this case is a relatively unique value. Say also that interval-anonymization without point suppression was applied for anonymizing the data. In that case an attacker can just look for trajectory groups, with attached demographic data intervals, over the city that he knows that the individual lives in, and search for intervals including the individual that is being attacked. If the individual's demographic data is sufficiently unique only a few sequences will be found and the attacker can strongly suspect that the individual that is being attacked has traveled on, at least, some of these few sequences.

When using the overlapping trajectory anonymizer, there is yet another disadvantage of using the interval-anonymization without point suppression on this problem, which is due to the fact that location data is considered as both quasi-identifier and sensitive value. Say that an attacker, for some reason, knows that a specific individual is leaving a specific station during a specific time interval (it might be possible to know this from surveillance cameras etc.). Say also that all possible trajectory groups which starts in this station is represented in the anonymized set, up to a node several routes away from the starting station. Now, there might be a risk that the individual that is being attacked is only included in one of the trajectory groups which the starts at the station (this depends on how unusual demographic data the attacked individual has). Then the attacker knows exactly which sequence the individual travelled on, because the individual cannot be included in any other sequence. This is clearly a serious inference attack (the attacker knows for sure where the individual went). Again, this problem occurs due to the fact that location data is both quasi-identifier and sensitive value.

How to anonymize demographic data linked to trajectories

How do we address the problems described above? Three possible solutions have been found. Each solution confuses a possible attacker in one way or another.

Attempt 1: One way is to add trajectories in order to confuse a possible attacker and then apply interval-anonymization without point suppression. For example, suppose that an individual starts a new sequence in station A and the demographic data is unique enough such that the demographic data is only included in one of the sequences' intervals in the anonymized set starting at station A. Say also that this sequence ends in station B. To confuse the attacker we can then add a new trajectory going from A to, another station, C. However, if this happens every weekday this new trajectory must be saved so that the next time this individual goes from A to B the other trajectory going from A to C is used again. If we don't do this, an attacker can easily see which trajectories that are real ones and which trajectories that are inserted to confuse the attacker because only one trajectory is consistent. It may also be possible that the individual takes the sequence B to A, say, eight hours after the initial sequence started. If a sequence from C to A does not occur at the same time the attacker will probably eventually conclude which of the trajectories is the right one. As you can probably see from this example is that it is very hard to prevent all kinds of inference attack by only adding new trajectories to the original data set. Also, this does not prevent a possible attacker from finding out that the specific individual frequently visits station A. One might even say that this makes it easier for an attacker to conclude this fact. Yet another problem with this approach is that it adds a lot of false data which will influence the quality of the anonymized data negatively.

Attempt 2: Another approach to solve this problem is to do the following. One can find the distribution of points in the demographic data to a specific sequence and then regenerate new points according to this distribution. If interval-anonymization without point suppression is applied to this new data it is very hard to conclude if a specific individual is in that trajectory group or not. Even if the individual's data is relatively unique and an interval is matching the individual's demographic data this could be due to a generated point and the individual was not in the corresponding trajectory group. Even though this might seem like a good approach in short-term it turns out that it is not a good approach in long-term if k is relatively low (10^1 in order of magnitude). This is due to the fact that we often have frequently occurring data, many people take the same trip to work every weekday and does so even the same time every day. If we look at the anonymized data from regenerated data of the same original data frequently, and there are not too many people who travels this specific sequence, it will not be long until we can make some conclusions of what the original data looks like. Therefore this approach is not sufficient. Also, this approach perturbs the data which, of course, results in higher information loss which is not desirable.

Attempt 3: The last possible approach that we came up with is to remove some data. One approach is to remove outliers in the demographic data. This eliminates the problem that intervals becomes unnecessary large when outliers are part of the set because all points must be included, which is a requirement for some approaches (such as interval-anonymization without point suppression). Another advantage of doing this is that an attacker cannot conclude that a certain individual did not travel a specific sequence, because the attacker does not know if the individual was an outlier and therefore suppressed from the output or if the individual did not travel that specific sequence. However, there are some ways to perform inference attacks which may work if only outliers are removed. The points on the boundary are especially prone to attacks. For example, say that we have person with the highest income in a neighborhood, without having that high of an income in order to be an outlier. If this person travels a sequence one day, this individual can easily be found, if the approach for anonymizing demographic data, sets the intervals such that the boundary points are prone to inference attack. An inference attack on boundary points can for example be possible if the interval boundaries are set exactly where the points on the boundaries are located. This means that the boundary of the intervals must somehow be set such that it does not depend on a single individual per interval boundary. Hence, even though the approach of removing some outliers can prevent some kinds of inference attacks, some other aspects needs to be accounted for as well.

We now have the rules required for the second part of the anonymization method.

4.4.1 Rules for demographic anonymizer

1. If an attacker knows the demographic data of a specific individual and also knows that this individual traveled a specific subsequence, the attacker should not be able to conclude the full sequence from the anonymized set (inference attack by exploiting that sequences are both quasi-identifier and sensitive value). This means that the attacker should not be able to link some external information with the anonymized set in order to find what sequence a specific individual travelled.
2. The anonymized interval boundaries must be set such that it is not possible to perform an inference attack using the positions of the boundaries.

Also, remember that what is anonymized by the method described below is the demographic data to individuals which belongs to *one* trajectory group at a time.

This method suppresses outliers, so the first rule is fulfilled. To prevent inference attacks, which takes advantage of the position of the boundaries, the resolution of the smallest interval of the anonymized set should not be too high (as was explained earlier). For example, the demographic data *age* can have intervals of five years as the smallest interval rather than one interval per year.

Because the intervals' boundaries are discrete and the resolution may be low, it is difficult to do inference attacks that takes advantage of that the second rule is not fulfilled. However, in order to completely fulfill the second rule *boundary hiding* should be implemented as well, which is a method described in Section 4.4.5.

The demographic data anonymization is a genetic algorithm and an introduction to this concept is given below. The approach is to minimize the information loss when the demographic data anonymization is applied. The definitions for the information loss is given in Section 4.4.3 and the algorithm to minimize this is described in Section 4.4.4.

4.4.2 Introduction to genetic algorithms

A genetic algorithm is a stochastic algorithm which is strongly inspired by Darwinism. The algorithm uses the approach of "survival of the fittest" on different solutions to distinguish the better solutions from the rest. Small changes are also applied as well as mixing different solutions to do a heuristic search of the space of solutions to find better and better solutions over time. It should be noted that this is only an introduction to genetic algorithms. Some parts of genetic algorithms have been left out for simplicity, and many variations of the methods described here exists as well.

A population of so called chromosomes are generated, where each chromosome represents a possible solution of the problem which we want to optimize. A fitness function must be defined, which evaluates how good a chromosome is. The goal of the genetic algorithm is to either maximize or minimize the value of the given fitness function. The initial chromosomes can be completely randomly generated. However, it is common to create rather good initial chromosomes through some simple function in order to speed up the evolution. These initial chromosomes should be different to some degree in order to have a sufficiently large search space to start with.

During each iteration, or so called generation, of the algorithm, chromosomes are selected using some selection method. There are many different selection methods but what they have in common is that better chromosomes have a higher probability of being selected. This is where "survival of the fittest" is applied. Each selected chromosome can then be both mutated and applied to crossover. A mutation is when a small change is done to the chromosome. Mostly mutations are bad (resulting in a worse fitness value) but a few is to the chromosome's advantage. A crossover combines the solutions of two chromosomes to create new offsprings, which are new chromosomes, which in turn replaces its parents.

Through these iterations, better and better chromosomes emerges when this heuristic search of the solution space is performed. The evolution is stopped when some criteria has been attained, for example after a fixed number of iterations. The chromosome with the best fitness value of all generations is the output of the algorithm.

Because a genetic algorithm is a stochastic algorithm, the output is not guaranteed to be optimal and the output differs for different runs.

4.4.3 Defining the ***Kullback penalty***

Several definitions of information loss when anonymizing data have been introduced in the literature [4, 2]. These have in common that they minimize the aspects of information that the author finds important. However, these definitions are often ad hoc and hence without theoretic foundation.

In this thesis a different approach is used, which, to the best of the author's knowledge, has not been used before. In the well known field of information theory there is a definition for information gain of an observation, the Kullback information, K. Our approach is to let the information loss, IL, from the original data to the anonymized set, be reflected by the information one would gain from first observing the anonymized set to getting access to the original data. The value of this information gain is given by the Kullback information.

The Kullback information has the following definition:

$$K\left[P^{(0)};P\right] = \sum_{i=1}^{n} p_i \log \frac{p_i}{p_i^{(0)}} \tag{3}$$

where $P^{(0)} = \left\{p_i^{(0)}\right\}_i^n$ and $P = \{p_i\}_i^n$.

For this definition we also define $0 \cdot \log(0/0) \equiv 0$.

$P^{(0)}$ is the *a priori* probability distribution of a system of which we do not have full knowledge and P is the probability distribution of an observation in which more knowledge is obtained. The Kullback information also requires that whenever the *a priori* probability $p_i^{(0)} = 0$ we also have that $p_i = 0$ (if $p_i^{(0)} = 0$ and $p_i > 0$ we would obtain infinite information from this observation).

We now define the information loss of going from the original set to the anonymized set as the Kullback information where the anonymized set is the precondition and the original set is the postcondition or, in other words:

$$o_s = \text{original set},\ a_s = \text{anonymized set} \tag{4}$$

$$IL(o_s \to a_s) \equiv K\left[a_s; o_s\right] \tag{5}$$

See Figure 5.

$$\begin{tikzcd} o_s \arrow[r, bend left, "IL"] & a_s \arrow[l, bend left, "K"] \end{tikzcd}$$

Figure 5: A schematic picture of how the information loss presented in this thesis is defined.

In order to calculate probability p_i of the original set, the points in the original set are first discretized into intervals. Exactly how the points are discretized is up to the implementer of the algorithm. Here, we will only use uniform interval sizes for each demographic data. This means that we are not computing the information loss from the actual original set to the anonymized set but from a discretized version of the original set, o'_s, to the anonymized set.

The smallest possible interval is given by the fixed interval positions in each dimension. These intervals creates small volumes in the full dimensional space. We call these volumes *atomic intervals*. A probability p_i is given for each atomic interval.

The output of the algorithm is d-dimensional intervals in the anonymized set, where d is the total number of dimensions. These intervals are non-overlapping and can be larger than the atomic intervals but the interval boundaries of the intervals in the anonymized set coincides with interval boundaries of the atomic intervals. In fact, each interval in the anonymized set, a so called *anonymized interval*, consists of one or more atomic intervals such that a rectangular cuboid in the entire dimensional space is formed. Hence, an anonymized interval is the equivalence to what was previously defined as *equivalence class*.

Each anonymized interval also has a number attached which represents the number of people in that interval. Because k-anonymization is applied, intervals with less than k people will have its attached number truncated to zero. See Figure 6 for an example of an output.

In order to simplify equations later on, we make the following definitions.

I_i	:	the i'th atomic interval	(6)
n	:	the total number of atomic intervals	(7)
$A(I_i)$	:	the anonymized interval that I_i belongs to	(8)
A_j	:	the j'th anonymized interval	(9)
$S(A_j)$	:	the number of atomic intervals A_j contains	(10)
m	:	the total number of anonymized intervals	(11)
D	:	the entire space in which all points can be located	(12)
$N(I')$	:	the number of points in interval I'	(13)
		I' can be either an anonymized or atomic interval	

p_i represents the probability in the atomic interval I_i for $o'_s$and $p_i^{(0)}$ represents the probability in the same atomic interval, I_i, for a_s.

The probability of p_i for $o'_s$is the probability of finding a specific point in I_i or, in other words, the number of points in I_i divided by the total

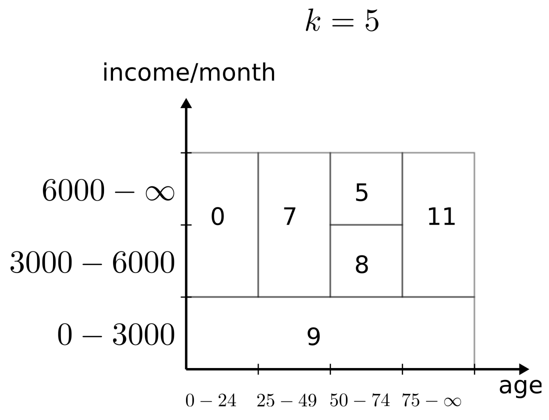


Figure 6: An example of the output of k-anonymization on strict multidimensional partitioning. The number in each anonymized interval represents the number of people with the corresponding demographic data.

number of points:

$$\boxed{p_i = \frac{N(I_i)}{N(D)}} \tag{14}$$

This, of course, has the property of probabilities that

$$0 \le p_i \le 1, \ \sum_i^n p_i = 1 \tag{15}$$

The probability $p_i^{(0)}$ of the anonymized set is somewhat similar. If there are more than k points in the anonymized interval that $p_i^{(0)}$ belongs to, this means that these points will be represented in the output of this algorithm. In that case, the definition of $p_i^{(0)}$ should be similar to the definition of p_i. However, the anonymized interval that $p_i^{(0)}$ belongs to can be larger than the atomic interval that p_i belongs to. Also, $p_i^{(0)}$ should represent the probability of finding a specific point in interval I_i. Therefore, we have to normalize $p_i^{(0)}$ such that it has the same value as all other probabilities in the same anonymized interval, or, in other words, $p_i^{(0)} = p_j^{(0)} \ \forall i, j$ where $A(I_i) = A(I_j)$. Because the interval sizes are uniform, we have

$$p_i^{(0)} = \frac{N(A(I_i))}{N(D)} \frac{1}{S(A(I_i))}, \text{ if } N(A(I_i)) \ge k \tag{16}$$

Intervals which contain less than k points are suppressed in the anonymized set. Therefore, if $N(A(I_i)) < k$ then $p_i^{(0)}$ equals the probability of finding a specific suppressed point throughout all intervals which are suppressed in the anonymized set. This reflects that every point which is suppressed in the anonymized set is indistinguishable from all other suppressed points.

This can be expressed as

$$I_{\text{suppr}} = \bigcup_i^n \left\{ \begin{array}{ll} I_i & \text{if } N(A(I_i)) < k \\ \emptyset & \text{otherwise} \end{array} \right. \tag{17}$$

$$p_i^{(0)} = \frac{N(I_{\text{suppr}})}{N(D)} \frac{1}{S(I_{\text{suppr}})}, \text{ if } N(A(I_i)) < k \tag{18}$$

To summarize the definitions above, we now have:

$$\boxed{p_i^{(0)} = \begin{cases} \frac{N(A(I_i))}{N(D)} \frac{1}{S(A(I_i))} & \text{if } N(A(I_i)) \geq k \\ \frac{N(I_{\text{suppr}})}{N(D)} \frac{1}{S(I_{\text{suppr}})} & \text{otherwise} \end{cases}} \tag{19}$$

By using this definition the following holds

$$0 \leq p_i^{(0)} \leq 1, \sum_i^n p_i^{(0)} = 1 \tag{20}$$

So, we want to find intervals for the anonymized set in order to minimize the information loss. In order to optimize the computation we rewrite the information loss:

$$\begin{aligned} \min\left(\sum_{i=1}^n p_i \log \frac{p_i}{p_i^{(0)}}\right) &= \min\left(\sum_{i=1}^n p_i \left(\log \frac{1}{p_i^{(0)}} - \log \frac{1}{p_i}\right)\right) \\ &= \min\left(\sum_{i=1}^n p_i \log \frac{1}{p_i^{(0)}} - \underbrace{\sum_{i=1}^n p_i \log \frac{1}{p_i}}_{C}\right) \end{aligned} \tag{21}$$

Because the original set is static, p_i is static $\forall i$ and thus C is a constant. Therefore we only need to minimize

$$\min\left(\sum_{i=1}^n p_i \log \frac{1}{p_i^{(0)}}\right) \tag{22}$$

To speed up the computation the equation above is rewritten such that we want to minimize

$$\boxed{\min\left(-\sum_{i=1}^n p_i \log p_i^{(0)}\right)} \tag{23}$$

(unnecessary division computation was removed).

If we look closely at equation (23) we are able to optimize the computation even more. We start by rewriting it as

$$\min\left(-\sum_j^m \sum_{i \text{ s.t.} I_i \in A_j} p_i \log p_i^{(0)}\right) \tag{24}$$

This is possible since anonymized intervals are non-overlapping.

If we look at equation (19) we see that $p_i^{(0)}$ is identical $\forall i$ s.t.$I_i \in A_j$.

We now define

$$p_{A_j}^{(0)} \equiv p_i^{(0)} \forall i \text{ s.t.} I_i \in A_j \tag{25}$$

Using equation (19) we have

$$\boxed{p_{A_j}^{(0)} = \begin{cases} \frac{N(A_j)}{N(D)} \frac{1}{S(A_j)} & \text{if } N(A_j) \geq k \\ \frac{N(I_{\text{suppr}})}{N(D)} \frac{1}{S(I_{\text{suppr}})} & \text{otherwise} \end{cases}} \tag{26}$$

Thus equation (24) can be rewritten as

$$\min\left(-\sum_{j}^{m}\left[\underbrace{\left(\sum_{i \text{ s.t.} I_i\in A_j} p_i\right)}_{P_{A_j}}\cdot \log p_{A_j}^{(0)}\right]\right) \tag{27}$$

$$P_{A_j} = \sum_{i \text{ s.t.} I_i\in A_j} p_i \overset{(14)}{=} \sum_{i \text{ s.t.} I_i\in A_j} \frac{N(I_i)}{N(D)} = \frac{N(A_j)}{N(D)} \tag{28}$$

$N(D)$ is a positive constant, so this can be removed when minimizing the function.

Thus, instead of minimizing equation (23), we minimize

$$\boxed{\min\left(-\sum_{j}^{m}\left[N(A_j)\cdot \log p_{A_j}^{(0)}\right]\right)} \tag{29}$$

As long as at least some of the anonymized intervals are larger than the atomic intervals, it is computationally faster to minimize equation (29) compared to directly minimize equation (23).

We call this function the *Kullback penalty*:

$$\boxed{KP \equiv -\sum_{j}^{m}\left[N(A_j)\cdot \log p_{A_j}^{(0)}\right]} \tag{30}$$

Another important fact from this sum is that

$$0 \le p_{A_j}^{(0)} \le 1 \forall i \Rightarrow \log p_{A_j}^{(0)} \le 0 \Rightarrow -\sum_{j}^{m}\left[N(A_j)\cdot \log p_{A_j}^{(0)}\right] \ge 0 \tag{31}$$

This is important because some implementations of genetic algorithms requires that the fitness value is greater than or equal to zero. Therefore, the Kullback penalty function can be directly used as a fitness function, if that is desirable.

Comparison with *Discernability metric* In [2] another cost metric to minimize information loss, for k-anonymization, was introduced. That is the *discernability metric*, which is a cost metric that can also be used on multidimensional splits.

$$C_{\text{DM}} = \sum_{j \text{ s.t.} N(A_j)\ge k}^{m} N(A_j)^2 + \sum_{j \text{ s.t.} N(A_j)< k}^{m} N(D)N(A_j) \tag{32}$$

However, as stated before, this metric is somewhat ad hoc, and used for its ability to "capture in a straightforward way the desire to maintain discernability between tuples as much as is allowed by a given setting of k" [2].

In comparison, the Kullback penalty, also has one more advantage compared to the discernability metric. The discernability metric minimizes the number of points in each anonymized interval. However, the size of each anonymized interval is not considered by the metric. Hence, the optimal output may contain large anonymized intervals where the points may be located in only small parts of these intervals. This aspect is captured by the minimization of the Kullback penalty because it, roughly speaking, maximizes the average density of non-suppressed intervals, while, at the same time, penalizes suppressed points. Thus, if a large anonymized interval has between k and $2k$ points, and all points are located in just a subinterval I' of this interval, a higher density is obtained if this subinterval is an anonymized interval by itself. This is captured by the Kullback penalty but not by the discernability metric. Similarly, if only most points, but still more than or equal to k points, are in I', in some cases the optimal configuration for the Kullback penalty is when the points outside I' are suppressed and I' is an anonymized interval. This depends on the density of the points in I' relative to the penalty which is given by the suppressed points.

Algorithm The Kullback penalty can be computed using algorithm 6.

Algorithm 6 Compute Kullback penalty

```
KullbackPenalty[root, k, D]
Require: $k \geq 2$
Ensure: $il \geq 0$
  $L_{\text{unsuppr}} \leftarrow \{l.A_j | N(l.A_j) \geq k,\ l \in root.leaves\}$
  $L_{\text{suppr}} \leftarrow \{l.A_j | N(l.A_j) < k,\ l \in root.leaves\}$
  $I_{\text{suppr}} \leftarrow \bigcup_{A_j \in L_{\text{suppr}}} A_j$
  $il_{\text{unsuppr}} \leftarrow -\sum_{A_j \in L_{\text{unsuppr}}} \left[ N(A_j) \log \left( \frac{N(A_j)}{N(D)} \frac{1}{S(A_j)} \right) \right]$
  $il_{\text{suppr}} \leftarrow -N(I_{\text{suppr}}) \log \left( \frac{N(I_{\text{suppr}})}{N(D)} \frac{1}{S(I_{\text{suppr}})} \right)$
  $il \leftarrow il_{\text{unsuppr}} + il_{\text{suppr}}$
  return $il$
```

4.4.4 Genetic algorithm approach for optimizing the Kullback penalty with k-anonymization as constraint

The problem described in Section 4.4.3 is NP-complete. This can be shown from the equivalence of the problem described in [6], which is also NP-complete. Therefore, an algorithm which does not necessarily find the optimal, but close to optimal solution was developed. A genetic algorithm was chosen because of its ability to find a near optimal solution in dimension spaces which may differ a lot for small changes of the input and has many local minima. However, as we will show later on, this system has a very erratic behavior which is not the perfect problem for a genetic algorithm. To account for this, only small modifications are made to the so called chromosomes which are used in the genetic algorithm.

Note that the method described in this section is a general anonymizer for d-dimensional points (where d is an arbitrary positive integer) and can be used to anonymize any kind of demographic data and does not depend on any method described in this thesis.

Because we want the output of the algorithm to be a set of rectangular cuboid intervals, we can perform splits on the full domain. Thus, a so called *binary space partitioning* is created. After one split in one dimension has been made, two new intervals have been created. These can, in turn,

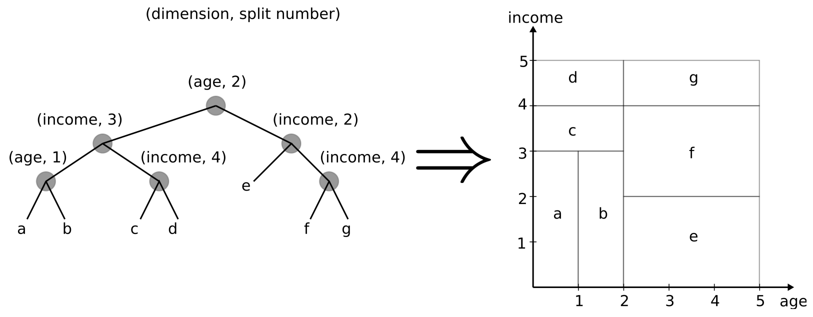


Figure 7: An example of a binary space partitioning tree together with the resulting binary space partitioning. The left child of a branch is the partition below the branch's split and the right child is the partition above the split. The leaves in the tree, represented by a to g, are represented by the corresponding partitions in the two dimensional space.

be split again, independent to each other. Thus, one of the two created intervals can be split multiple times while the other remains as a full interval. These splits can occur until the smallest possible intervals, the atomic intervals, emerges.

This will create so called *strict multidimensional splits* [6]. These splits creates the anonymized intervals of the anonymized set.

These splits are represented by a special case of a *binary space partitioning tree*. Each branch in the tree contains a split, which is represented by the dimension of the split together with where, in this dimension, that the split occurs. Each branch also contains the interval that it represents. The left child of a branch is the interval below the split in the branch and, respectively, the right child is the interval above the split in the branch. If a child is a branch this interval is split even further. The leaves are empty and shows that no more splits occurs and each leaf is therefore one of the anonymized intervals which the full tree represents. See Figure 7 for an example of a tree together with the resulting intervals.

Each chromosome in the genetic algorithm contains a valid tree. A valid tree means that, for each branch, the split in the branch is not on the boundary or outside its own interval.

For example, a tree has two splits, represented by the root node that is a branch and its left child which is also a branch. All other nodes are leaves. The left child must, in this case contain a split in its own interval, that is, in the lower interval that is given by the root branch's split. The left child can thus not have a split in the interval represented by the root's right child or have the same split as the root branch.

Mutations The tree has a very erratic behavior. A small change in one of the top branches can have a huge impact on the resulting intervals. Therefore the mutations are designed to make small, but still noticeable changes.

Four different mutations were developed:

Split mutation: This mutation starts with finding all leaves which consists of anonymized intervals containing more than one atomic interval. These leaves can be divided. We call the set of these leaves L_{split}. Each leaf in L_{split} is then selected for mutation with a probability of $1/|L_{\text{split}}|$. The decision of applying a mutation is done for each leaf in L_{split} independently. Thus zero to $|L_{\text{split}}|$ mutations can occur if this mutation is applied. Splits are performed on all leaves that were selected. The performed split is one of all possible splits that can occur on the leaf, where all possible splits in the leaf have the same probability of being performed.

This mutation can be considered as the opposite to the merge mutation, as described next.

Merge mutation: This mutation can merge some branches to a leaf. All parents which has at least one child which is a leaf creates the set P_{merge}. After that, each of the branches in P_{merge} is selected with the probability of $1/|P_{\text{merge}}|$. As with the split mutation, the decision for mutating a branch is independent from all other branches. If a mutation occurs it simply converts the selected split branch into a leaf (and thereby merges its children's intervals into the interval in the selected branch). Again, the reason why only parents which has at least one child which is a leaf may be mutated by this mutation is because the mutations should not have a too high impact on the chromosomes.

This mutation can be considered as the opposite to the split mutation (the mutation above).

Move split mutation: This mutation starts with selecting one of all branches in the tree, where each branch has equal probability of being selected. The split of the selected branch is moved one atomic interval, in the same dimension, to either one atomic interval level above or below. The direction of the mutation (above or below) has equal probability.

If the split is moved "outside" the valid range of splits, that is, it is moved to either one step above the highest valid split or one step below the lowest valid split, the branch is removed. If the split is moved one step above the highest valid split, the left child, which is the child with the lower interval, replaces the branch on which the mutation occurred. Similarly, if the split is moved one step below the lowest valid split, the right child, which is the child with the higher interval, replaces the branch on which the mutation occurred. See Figure 8 for an example when a split is moved "outside" the valid range of splits.

After the mutation it might happen that the tree is no longer valid and has to be fixed. This happens when the split is moved to a split which occurs on a branch below the the mutated split. The split, on the branch below the mutated split, is now either one step above the highest valid split or one step below the lowest valid split. To fix the tree a simple search for all branches below the mutated branch, with the same split as the mutated branch, is performed. These branches are simply removed from the tree (converted to a leaf) and the tree is yet again valid.

Branch swap mutation: Just as with the move split mutation, this method starts by selecting one of all branches in the tree, and each branch has equal probability of being selected. However, the root node cannot be selected by this mutation. The mutation then swaps the position of the selected branch, b, with its parent p. When doing so, one of b's children, c_i, must be replaced by p. However, p has a vacant position, where b originally was. Thus c_i is placed in this position, such that c_i becomes one of the two children of p. The child c_i of b is selected at random, with equal probability of the two children of b. See Figure 9 for an examples of this mutation.

Similar to the move split mutation, this mutation can result in a non-valid tree. This time we have to perform a full fix on the parent branch of the b branch *after* the mutation. This is done as follows: In each branch, check if the split in the branch is valid for its interval. If it is not valid,

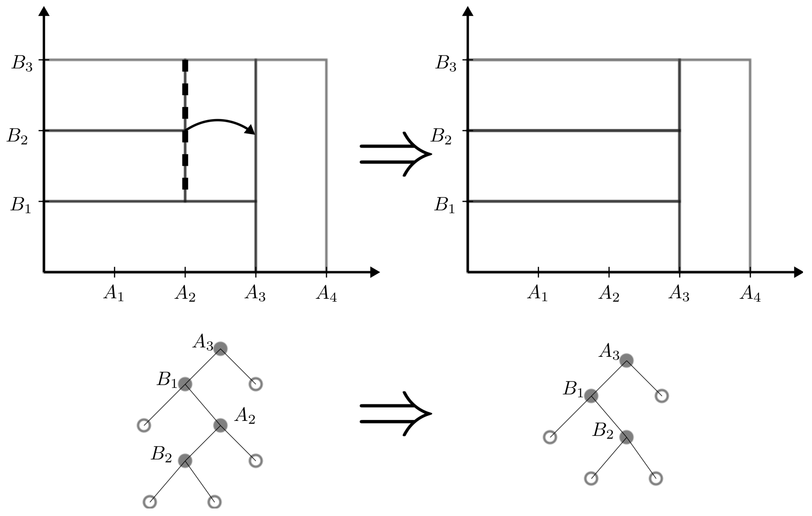


Figure 8: An example of a move split mutation with both the two dimensional space and the corresponding tree. The branch is, in this case, moved outside the valid range, so that the branch is replaced by its corresponding child.

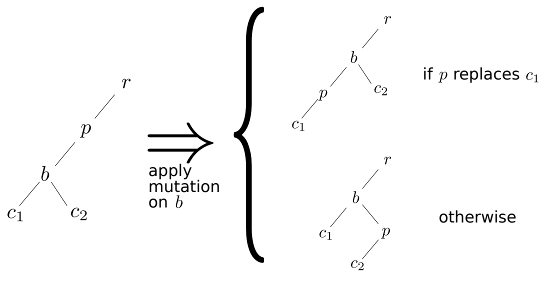


Figure 9: An example of a branch swap mutation. The branch b is randomly chosen as the branch to mutate. One of the two possible branch swap mutations for b is applied (where both possible mutations have equal probability of being performed). The leaves are not shown in this figure.

convert the branch to a leaf. If it is valid, perform the split in the branch and set the resulting intervals for both children correspondingly and continue the same procedure on the children. Because all branches which may not be valid (after the mutation) are checked and fixed, the resulting tree is valid.

This mutation might seem somewhat arbitrary, but there is a good reason for why it is used. If we have a tree, where the lower branches have splits which results in a low Kullback penalty, but some of the branches in the upper part of the tree are not optimal, this mutation can help the chromosome from getting stuck in a local minima by swapping branches in the upper part of the tree.

Crossover Only one type of crossover is used and it is the type of crossover that is usually used on binary trees.

The crossover goes as follows. Two trees are randomly selected for crossover. On each tree a cut between a random child and its parent node is performed. The subtree below the cut of one tree is combined with the subtree above the cut of the other tree. This creates one chromosome. If vice versa is done, we can also get another chromosome from the crossover at the same time.

This can, of course, create invalid trees, so the new trees must be fixed before they are returned as a chromosomes. This fix only needs to be performed on the subtree of the parent branch of the cut, because the rest of the tree should be valid. See Figure 10 for an example of crossover.

Initial chromosomes In order to speed up the search for the optimal solution, using a genetic algorithm, it is common to help the evolution by creating initial chromosomes which are not completely random, but rather have a good approximation of the optimal solution.

The initial chromosomes are created using a recursive function, where the root branch of an empty tree is the input parameter of the first call to this function. In each call, the function searches through all possible splits which can be performed on the current interval, to find the splits which divides the current branch such that at least k points are in each of the two resulting leaves. One of these splits is selected at random and is then performed on the branch. The children, given by the split, are, independent to each other, divided through the same recursive function. Hence, no points are suppressed using this method. If no splits, as the ones described above, can be found, the current branch is not divided. Hence, the recursive function is guaranteed to terminate.

Furthermore, a few completely random valid trees are also generated for the initial chromosomes to get a better initial cover of the search space.

4.4.5 Boundary hiding

When implementing the demographic data anonymizer described in Section 4.4.4, the resulting intervals represent the number of people in each interval. From these intervals we can strongly suspect that there is at least one person represented in one atomic interval on the boundary of an anonymized interval. Even a single person can make a difference of where the boundary is and therefore boundaries might be prone to inference attacks. Say that a possible attacker knows that a person, p, living in an area, has a relatively unique demographic data in one attribute. Now, if the attacker finds out that this data is represented in a trajectory

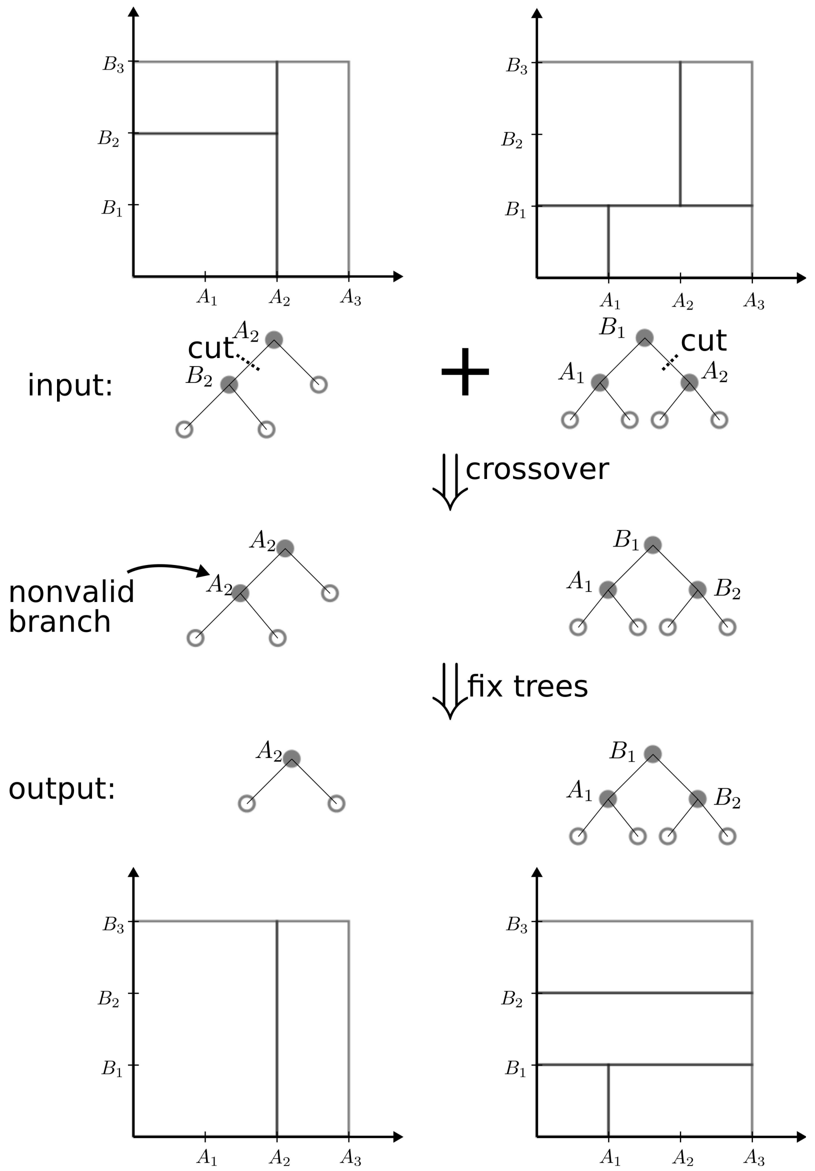


Figure 10: An example of a crossover of two binary space partitioning trees. The left resulting tree is nonvalid and is fixed before it is returned as an output. The input and output partitionings are also shown in the figure. A and B are the two dimensions and the subscript of each dimension is the dimension split.

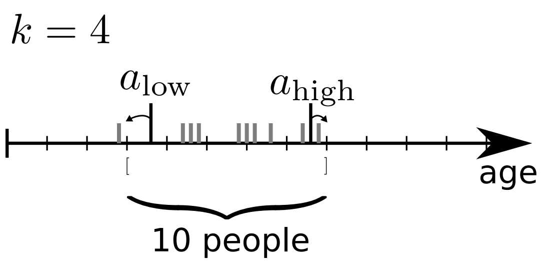


Figure 11: An example where boundary hiding is applied. The grey lines represents the age of different individuals. $k/2 = 2$ so the average value of the two individuals with lowest age is computed and stored in the variable a_{low}. The lower boundary is then moved to the lower coarse grained value (atomic interval) closest to a_{low}. The corresponding is done to the two highest values and the higher boundary is moved to the higher coarse grained value (atomic interval) closest to a_{high}.

group leaving from this area he can strongly suspect that p travelled on this subsequence and an inference attack has in that case been performed. However, if it is possible to hide where the boundary is, in the case when there are few people in the trajectory group, it becomes very hard for a possible attacker to perform this kind of inference attack.

The idea of boundary hiding goes as follows. The boundary is hidden in one dimension at a time. The average value of the $k/2$ lowest values in the anonymized interval is computed. We call this value a_{low}. The lower boundary is now moved to the closest lower atomic interval boundary to a_{low}. Respectively, the average value of the $k/2$ highest values is computed and is saved in the variable a_{high}. The higher boundary is then moved to the closest higher atomic interval boundary to a_{high}. However, even though the boundary has been moved we still say that the same number of people are in this new interval (the same number of people as in the original anonymized interval). See Figure 11 for an example of boundary hiding.

By applying this method it is hard to know exactly where the boundary is which prevents an attacker to perform inference attacks on boundaries. The reason for this is because a single point (representing demographic data for an individual) never decides exactly the boundary is (at least for $k > 2$, which we assume is always the case). However, because the change is very small, if there is a change at all, an analyzer of the data will probably not care about the small differences and will just consider the intervals' boundaries as the true intervals.

For a large number of people this might seem unnecessary because the probability of the kind of inference attacks that this method prevents is, in that case, very low. However, if this additional method is applied when it actually is unnecessary it will probably not change the data at all, and, if it does, the change is very small. Thus, this method only makes larger changes when it really is necessary. Also, this method is fast to execute so it is not a performance issue.

4.5 Additional methods

4.5.1 Interval size I_s

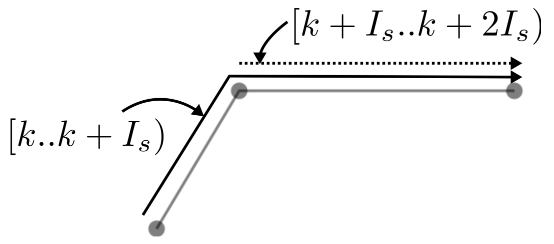


Figure 12: An example of two overlapping trajectory groups. The circles represents nodes and the grey lines are routes. The interval represented by $[a..b)$ is the interval of number of people in the corresponding trajectory group. The full line is thus a trajectory group which sequence consists of two routes and the dotted line is another trajectory group which sequence consists of one route.

I_s for the overlapping trajectory anonymizer As discussed before, in order to use the *overlapping trajectory anonymizer* the *interval filter* must be applied. The interval size I_s must be decided and to keep as much information as possible we want to minimize I_s. The following examples discusses the minimum size of I_s that is needed in order to prevent inference attacks.

Interval size example: Figure 12

In this example we see that there are more people traveling along the dotted subsequence compared to the full line. We also see that the minimum difference between the number of people is $\min([k + I_s..k + 2I_s)) - \max(k..k + I_s)) = 1$ and the maximum is $\max([k + I_s, k + 2I_s)) - \min(k, k + I_s)) = 2I_s - 1$. In other words, the difference in the number of people is $[1, 2I_s)$. If $I_s = k/2$ we obtain the difference in the number of people $[1..k)$. This is, strictly speaking, an inference attack according to k-anonymization because we might say something about an individual with higher probability than $1/k$ (the probability in this case is $1/(k-1)$). However, we can redefine what is considered an inference attack and instead say that if we know something about $k - 1$ individuals, or larger interval, it is not an inference attack. If $I_s < k/2$ even this new definition is violated, so it is required that $I_s \geq k/2$.

Hence, the anonymization used for *overlapping trajectory anonymizer* is:

k_{-1}-anonymization: $k_{-1} \equiv k - 1$
The anonymization is violated if we can say something about an individual with a probability higher than $1/k_{-1} = 1/(k - 1)$

Interval example: Figure 13

As is shown in the figure, the interval size $I_s = k/2$. In this example there is a difference between the intervals in the demographic data of two overlapping subsequences. This means that there must be a difference in the people that travelled along each subsequence. The dotted subsequence is completely overlapped by the full lined subsequence and therefore we can infer that the difference between the subsequences is that there are more people on the dotted subsequence compared to the full line. From the demographic data we can infer that at least one of the people that are only on the dotted subsequence has an age in the interval [30..35). From this information we can conclude that one to four ($[1..k/2)$) people travelled on

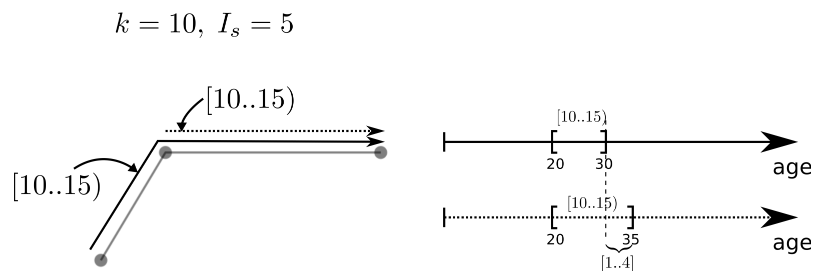


Figure 13: An example of two overlapping trajectory groups. The circles represents nodes and the grey lines are routes. The interval represented by $[a..b)$ is the interval of number of people in the corresponding trajectory group or demographic interval. The full line is thus a trajectory group which sequence consists of two routes and the dotted line is another trajectory group which sequence consists of one route. The two graphs to the right are the demographic intervals of the two trajectory groups. The interval in full line is the demographic interval for the trajectory group in full line and, correspondingly, the dotted interval is the demographic interval of the dotted trajectory group.

the dotted subsequence and has an age in the interval [30..35). Even if we use k_{-1}-anonymization, this is still considered an inference attack. If this kind of inference attack happens on multiple different demographic data at the same time, one might be able to strongly suspect that a specific person travelled along the dotted trajectory group.

To prevent these kinds of inference attack the interval size has to be larger than or equal to k. However, one might consider that interval size too large. An alternative approach is to accept these inference attacks, when the interval size is $k/2$ and set the value of k such that $k/2 - 1$ is "high enough". What value k should have in this case for $k/2 - 1$ to be "high enough" is up to the user of the algorithm.

If k is low we can extend this kind of inference attack even further. Say, for example that the demographic data *age* is divided into 20 intervals and that $k = 10$. If this extension of intervals (in the example above), between two different subsequences, occurs on roughly k different intervals in the same demographic data, at the same time, we know that there is roughly one individual in each extended interval. Even if this is unlikely to happen, this is a serious inference attack (we know something about a single individual) and thus k should be higher than the maximum number of divisions of any demographic data.

Number of people example: Figure 14

Another serious inference attack have been found when using this algorithm. However, this inference attack requires that the attacker has external information that is outside the scope of publicly available demographic data connected to corresponding individuals. Hence, the user of this algorithm might not consider this kind of inference attack.

This inference attack requires that the attacker knows exactly how many people that travelled on some subsequences. This data can, for example, be obtained from surveillance cameras. In Figure 14 we have an example where this kind of inference attack occurs. From the anonymized data alone we cannot perform an inference attack. However, let's first start with what information we can obtain from this figure before performing the inference attack where we know exactly how many peo-

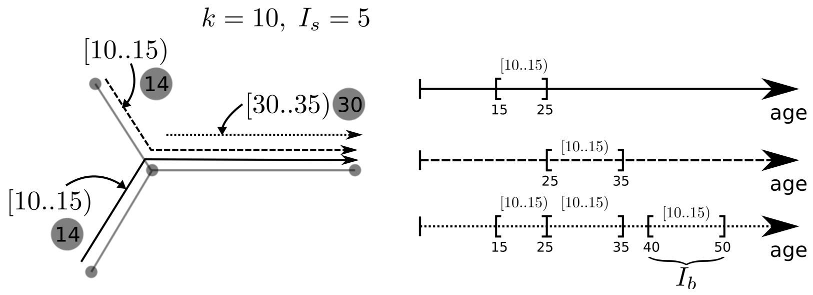


Figure 14: An example of three overlapping trajectory groups. The circles represents nodes and the grey lines are routes. The interval represented by $[a..b)$ is the interval of number of people in the corresponding trajectory group or demographic interval. The grey circles with a number is the exact number of people in the corresponding trajectory group (this is not shown in the anonymized set, but may be obtained from, for example, surveillance cameras). The three trajectory groups, the full line, the dotted and the dashed line, have corresponding demographic intervals to the right with the same shape.

ple that travelled on some subsequences. We know that there are people that travelled only on the dotted subsequence because, if we add up the maximum number of people on the full lined and dashed subsequence this number does not add up to the interval of the dotted subsequence, $\max([10..15) + [10..15)) = 14 + 14 < \min([30..35))$. We can also see that there is a demographic interval in the dotted subsequence that does not show up in the either the full lined or dashed subsequence. We call this interval I_b. The highest possible number of people in I_b, that comes from the full lined and dashed subsequence, is $4 + 4 = 8$, because the lowest possible number of people in the intervals that are shown in the full lined and dashed subsequences is ten respectively $(\max([10..15)) - \min([10..15)) = 4)$. Therefore at least two people from the dotted subsequence are represented in I_b.

So, if we now, for example, are able to obtain that 14 people travelled on the full lined subsequence and 14 on the dashed subsequence we have enough information to perform an inference attack. We can now calculate how many people that were at most present only on the dotted subsequence: $\max([30..35)) - (14 + 14) = 6$. If we add up this with our previous result we now know that [2..6] people travelled only on the dotted subsequence and [2..6] of these are represented in I_b. This is clearly an inference attack because $6 < k = 10$. If the attacker also is able to obtain that there were, for example, exactly 30 people who travelled along the dotted subsequence, we now know that there were only two people who travelled only on the dotted subsequence and that these two are represented by the interval I_b. If this happens on multiple demographic data it might be relatively easy to identify an individual.

It is also possible to show that this kind of inference attack can still be performed if we increase I_s to k.

Hence, by only obtaining exactly how many people who travels along some subsequences it can be possible to, together with the anonymized data, in some worst-case scenarios, identify individuals when using this algorithm, no matter how high the value of k is.

If k is high it might be harder for the attacker to know exactly how many people that travelled on a specific subsequence, even though it is of course still possible.

I_s for the nonoverlapping trajectory anonymizer When anonymizing using the nonoverlapping trajectory anonymizer, overlapping trajectory groups (with respect to sequences) may not include the same individual. I_s for the overlapping trajectory anonymizer was introduced because that method may construct overlapping trajectory groups which contains the same individual, which may lead to possible inference attacks if the value of I_s is not high enough. Thus, the nonoverlapping trajectory anonymizer together with boundary hiding has no need for the interval filter for the anonymization discussed so far. However, if some extra requirements are added, the interval filter may need to be applied to the output of the nonoverlapping trajectory anonymizer as well, see Section 4.5.2 and 4.5.3.

4.5.2 Concurrent release of overlapping data

When anonymizing demographic data, the data can be partitioned in different ways. One way is to group all demographic data of an individual to a d-dimensional point, where d is the number of demographic attributes. d-dimensional equivalence classes are then created which represents the anonymized set. Another approach is to divide each demographic data into d separate points, one for each demographic attribute. The anonymized set then contains separated equivalence classes, one for each dimension. A third approach is to have something in between, that is, group demographic attributes where one wants to see the correlation and separate groups where the correlation is not so important. For example, say that we have the three attributes age, gender and income, and we want to see the correlation between age and income, and we separately want to see the distribution of the gender. In that case we can separate the data into (age, income) and (gender) and the resulting equivalence classes in the anonymized set contains equivalence classes with these attributes, grouped in this way.

If several of these different types of equivalence classes are applied to the same data and then released, one might think that by combining the overlap between different equivalence classes, it is possible to perform an inference attack. This is often the case, but in our case it is required that the interval size $I_s = k$ for the overlapping trajectory anonymizer. This means that it is, in this case, not possible to infer the demographic data of less than k people. Hence, this does not violate k-anonymization. Thus, all possible combinations of anonymized versions of demographic data can safely be released concurrently without violating the privacy. This data can be used in several ways. If only one demographic data is anonymized, the highest possible resolution for this data is acquired. But in this case you cannot see the relation to other demographic data. If *all* demographic data is anonymized in the same set at the same time, the relation between all demographic data is obtained, but in this case the resolution will be low unless the number of data points is very high compared to k. If only the relation between some demographic data is desirable, the resolution will be somewhere in between the two extreme cases. Again, all the anonymized versions of combinations of the data can be released and an analyzer of the data can get exactly the combination (and relations) of demographic data he or she wants, with the highest possible resolution, while still preserving the privacy.

Again, this requires that $I_s = k$. In order to be able to do this when using the nonoverlapping trajectory anonymizer (which does not initially require to use the interval filter) we must set $I_s = k$ as well.

4.5.3 Time intervals

There are two alternatives to include the time when anonymizing the data.

The first alternative is to have fixed time intervals in which all trajectories to the input of the anonymization occurred. By using the same reasoning as in Section 4.5.1 (where the size of I_s for the overlapping trajectory anonymizer was concluded), it is possible to have overlapping time intervals, with respect to anonymized releases, if the interval filter is applied and I_s is sufficiently large (see Section 4.5.1). This can advantageously be used to see the movement only during rush hours and also be able to use the same data to see the movement of all times during a week. If interval filter is not applied (which only holds for the nonoverlapping trajectory anonymizer when boundary hiding is applied) only nonoverlapping time intervals, with respect to anonymized releases, are allowed.

The other alternative is to let the time be an attribute in the demographic data. Again, if the interval filter is not used, the same trajectory may not be used for multiple anonymizations.

4.5.4 Active inference attacks

Definition: An *active inference attack* occurs when one actively changes the raw data, which will be anonymized at a later time, in order to be able to perform an inference attack using both the exact information of the changed data, together with the anonymized data.

In the case for the problem described in the paper, active inference attack would be to travel a certain sequence with mobile phones registered with the same specific demographic data in order to elicit demographic data of other individuals traveling the same sequence. The attacker has now knowledge both about the exact demographic data of the mobile phones *and* the anonymized data of the sequence that he or she travelled. By combining this information the attacker might be able to identify an individual.

However, when we require that the interval filter is applied where $I_s = k$, active inference attacks are very hard to perform. If a possible attacker has $k-1$ mobile phones registered to the same demographic data and $[k..2k)$ users on this sequence are represented in the anonymized set, all the attacker knows is that $[1..k]$ users, who travelled on this sequence, has this demographic data. This is not a violation of k-anonymization.

There is one extreme case where this type of inference attack can be performed. Assume that an attacker has $k-1$ registered mobile phones, with the same demographic data, and travels on a sequence where n other people travelled. If $n < k$ then $[k..2k)$ people will be represented in the anonymized set as the number of people who travelled this sequence. If the demographic data, that were registered on the attackers mobile phones, are represented in the anonymized set, the attacker knows that at most n people has this demographic data. Because $n < k$ the k-anonymization has been violated. However, this type of inference attack is both time and resource consuming and we highly doubt that anyone would ever consider this just to suspect that a certain individual travelled

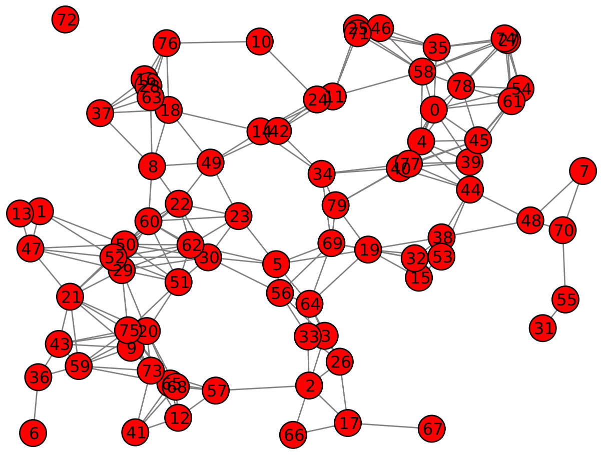


Figure 15: A randomly generated city which is used for evaluating both trajectory anonymization methods. A circle represents a node and these are connected by routes which goes in both directions.

on a specific sequence with a probability above $1/(k-1)$. To actually identify an individual with a high probability using this method requires a worst-case scenario which is very unlikely.

When the interval filter is not required to be applied it is obviously easier to perform active inference attacks. A user of the algorithm might thus consider implementing the interval filter with the only purpose of preventing active inference attacks. Even small values for I_s might be better than not using the interval filter at all when considering active inference attacks.

5 Experimental evaluation

5.1 Trajectory anonymization

The execution times of the nonoverlapping trajectory anonymizer and the overlapping trajectory anonymizer were compared using randomized "maps" which were generated as follows: The map on which the two methods are compared to each other is randomly generated using the following method. 80 nodes are uniformly distributed in the unit square and routes are created between all nodes with a distance of maximum 0.17 from each other. See Figure 15 for an example map. A number of trajectories were then randomly generated on this map. These trajectories has a length between four and 15 routes. A generated trajectory starts at a random node and performs a random walk to neighboring nodes (connected by routes) such that the same node is not visited several times for the same trajectory.

The implementation of both the nonoverlapping trajectory anonymizer

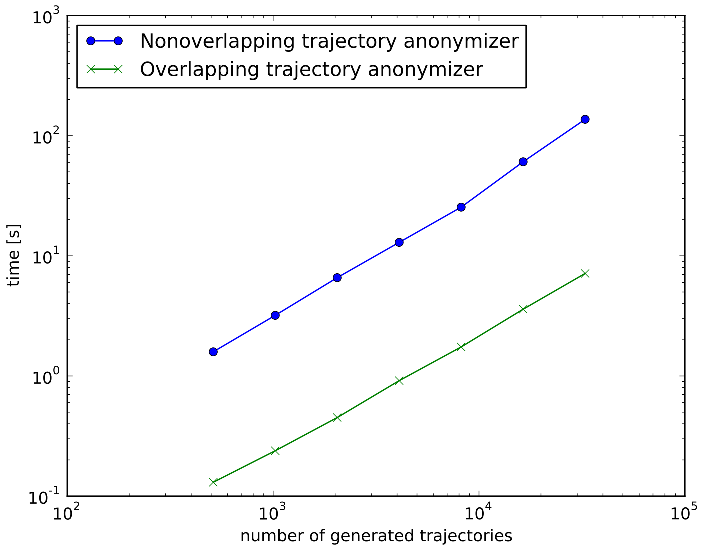


Figure 16: Benchmark times for executing the overlapping as well as nonoverlapping trajectory anonymizer. The overlapping trajectory anonymizer clearly outperforms the other method in execution time by at least one order of magnitude. This plot was obtained for $k = 10$. Each point in this plot is the average value of the output from five runs on five different randomly generated maps.

and the overlapping trajectory anonymizer were done in Python 2.6.6. All experiments were performed on an Intel(R) Core(TM) i5 CPU M 480 @ 2.67GHz with 3.5GB Memory. In Figure 16 the execution times for the two methods are presented. From this figure we see that the overlapping trajectory anonymizer clearly outperforms the nonoverlapping trajectory anonymizer. It is not surprising that the nonoverlapping trajectory anonymizer is slower than the overlapping trajectory anonymizer because the steps of the overlapping trajectory anonymizer is done as just a part of the entire nonoverlapping trajectory anonymizer.

A comparison of how much information which is preserved was done as well. This comparison was done on the same map which was generated to benchmark the execution times. We now make the following definitions. $l(t_i)$ is the length of the sequence of input trajectory number i. $l(tg_j)$ is the length of trajectory group number j and $s(tg_j)$ is the number of individuals represented in that trajectory group. The information which is preserved is reflected by the following ratio:

$$R = \frac{\sum_i l(t_i)}{\sum_j l(tg_j)s(tg_j)} \tag{33}$$

Thus, this ratio R reflects how much of the input trajectories that are represented in the anonymized set. Because the nonoverlapping trajectory anonymizer does not allow overlapping trajectory groups to include the same individual, this number is between zero and one. In contrast, each individual can be represented in overlapping trajectory groups for the overlapping trajectory anonymizer. Thus, the number for that method

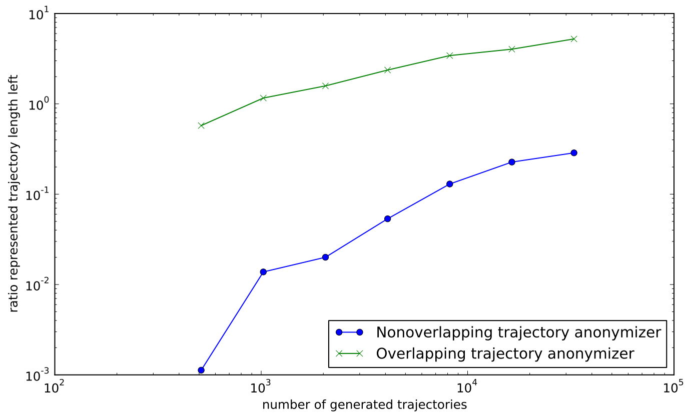


Figure 17: The x-axis is the number of generated trajectories in the city. The y-axis represents the ratio, between the total length of all anonymized trajectory groups times their corresponding number of individuals, in relation to the total length of all input trajectories. This plot was obtained for $k = 10$. Each point in this plot is the average value of the output from five runs on five different randomly generated maps.

can be larger than 1. The result can be seen in Figure 17. The fact that the overlapping trajectory anonymizer outperforms the nonoverlapping trajectory anonymizer is obvious.

5.2 Demographic data anonymization

This section evaluates the genetic algorithm which is used as demographic data anonymizer in this thesis. The data that was evaluated is the Adults database from the UC Irvene Machine Learning Repository [10], which is very common to use to benchmark k-anonymization methods [4, 2, 6]. The evaluation of the genetic algorithm presented in this thesis is compared to the greedy method described in [6]. The evaluation for both methods were done using the Kullback penalty and the discernability metric.

There are multiple articles which shows how to optimize micro data in so called full-domain generalization [5, 12] and single-dimensional partitioning [2, 4]. In [6] it is shown that the greedy algorithm in the same article results in higher quality, for real world experiments, compared to the optimal solutions using either full-domain generalization or single-dimensional partitioning. Also, the output of our algorithm produces multi-dimensional partitionings, just as in [6]. These are the reason to why we choosed to compare our algorithm with the one described in [6].

The configuration of the data was done as described in [6]. Thus, only eight attributes were used and the data does not have a categorical structure but instead all values are placed in a continuous ordering. The order of educational data is the same as in [4]. Only tuples, in the Adults dataset, with no missing values in the eight anonymized attributes were used. This results in 30,162 records.

In this thesis, the methods are not allowed to place points which are on the same boundary in different equivalence classes. However, this is allowed when executing the method we call Mondrian, which is the greedy method described in [6]. Hence, a few modifications were done to Mondrian in order to compare it with the genetic algorithm presented in this thesis. When choosing the dimension where the split will be performed in each iteration of Mondrian, the dimension with the widest normalized range of values is selected, just as in [6]. However, if the split given by that dimension results in that fewer than k points are in one of the two resulting partitions, this split is reverted. In that case, the dimension with the second widest normalized range of values is selected instead. This iteration continues until a dimension which split results in that the two resulting partitions has more than or equal to k points each is found, or until all dimensions have been evaluated, and failed to perform a valid split (a valid splits is when $\exists \geq k$ points in both resulting partitions). In the latter case the equivalence class is not divided.

For the genetic algorithm, the population is 40 and the number of generations is 40 as well. Crossover was applied with a probability of 0.8 and each mutation was also applied with a probability of 0.8. Elitism was used as well, where the best chromosome from each generation is directly moved to the next generation without applying any mutation.

Both resulting metric penalties for both the genetic algorithm and Mondrian can be seen in Figure 18 and 19 respectively. From these figures we see that the genetic algorithm performs better for both metrics and for all values of k which were evaluated. However, the difference when optimizing the discernability metric is very small. The reason for why there is a larger difference in the case for the Kullback penalty is due to the fact that Mondrian only optimizes to include as few points $\geq k$ in each equivalence class but does not try to optimize the density in all equivalence classes, which the genetic algorithm is able to do. However, because the genetic algorithm takes longer time to execute compared to Mondrian, Mondrian can be a good alternative to minimizing the Kullback penalty if the execution time is considered to a larger degree.

6 Discussion and future work

The problem for this thesis is somewhat different from most articles in the area of anonymization so far, because that the data that is anonymized contains both sequences travelled by people *and* their demographic data. Since the input contains different types of data, that is, sequences of data (representing travelled trajectories) as well as d-dimensional single points (representing demographic data), the anonymization was divided into two different major anonymization methods. However, because there is an indirect relation between the trajectory groups created by the first part of the main anonymization, the anonymization of the demographic data (which does not account for this indirect relation) does not guarantee the preservation of the privacy. Therefore, two additional filters were introduced in order to account for this indirect relation and hence preserve the privacy. If, however, a universal method, for anonymizing all data at once, is developed, then it is likely that no additional filters have to be applied. However, it is probably difficult to develop such a method since the sequential data and the d-dimensional single points are represented in very different ways.

Two different methods for anonymizing the trajectories were developed. Even though they have the same type of output (anonymized set containing trajectory groups) the actual output of the full anonymization has some differences. The output of the overlapping trajectory anonymizer

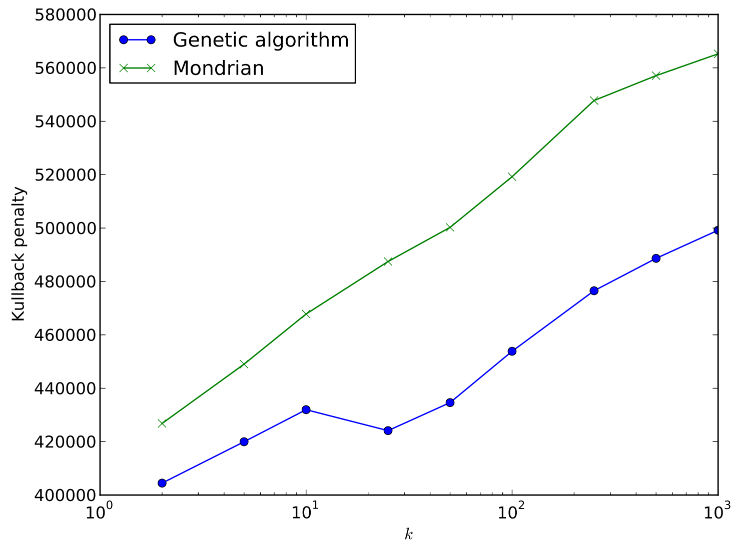


Figure 18: Resulting values for the Kullback penalty using the genetic algorithm and Mondrian. Every value for the genetic algorithm were obtained from one run each.

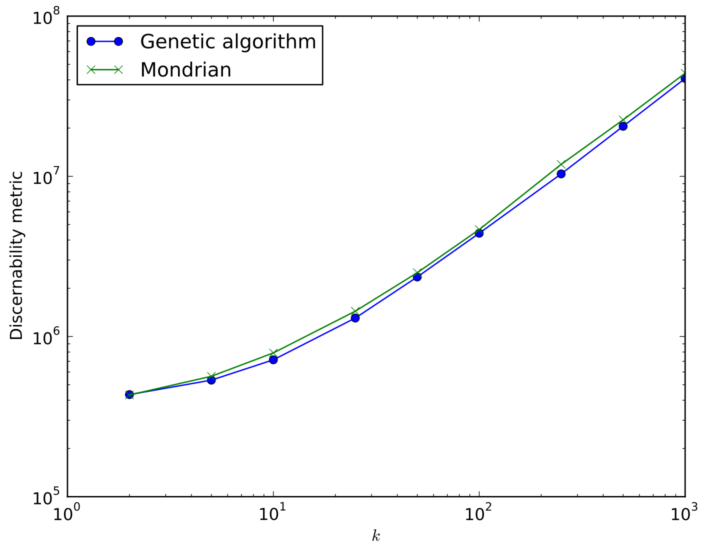


Figure 19: Resulting values for the discernability metric using the genetic algorithm and Mondrian. Each value for the genetic algorithm were obtained from one run. Even though the difference is small, the genetic algorithm has a more optimal (smaller) value than the results from Mondrian in all cases.

may include overlapping trajectory groups which contains the same individual. Because this is not allowed in the nonoverlapping trajectory anonymizer, there is a big difference in how much information which is preserved when comparing both methods. This is clearly shown in the experimental evaluation where the output of these two methods are compared. However, because the interval filter must be applied to the output of the overlapping trajectory anonymizer, where the recommended value for I_s is k, a big amount of information is unfortunately lost here. This is only a requirement for nonoverlapping trajectory anonymizer in some cases. Thus, the information loss is applied on different aspects, either on which individuals who are included in different trajectory groups, or in the certainty of the number of people.

The performance of the genetic algorithm presented in this thesis can probably be improved even more. For example, it might be possible to identify mutations to the chromosomes which results in better search heuristics. A more thorough search for better parameters of the genetic algorithm can be done as well.

7 Conclusions

In this thesis, two different methods are presented for anonymizing trajectories travelled by people, with associated their demographic data. One of the methods has low information loss but can, in some worst-case scenarios, be prone to inference attacks. The other method has a higher information loss, compared to the first one, but no possible inference attack has been found for this method. The anonymization consists mainly of two parts for both methods, where the second part is the same. The first part anonymizes only the trajectories while the second part anonymizes the demographic data of the individuals. The second part also makes sure that the output of the entire anonymization procedure is anonymized. k-anonymization, which is widely used for anonymizing spatio-temporal data [9], was used in both parts for both methods.

To only use a straightforward k-anonymization on two different methods, first on the travelled sequences and then on the demographic data, was not enough to preserve the privacy. Some additional filters were added to prevent the kinds of inference attacks that were identified after only the straightforward k-anonymization was applied. One of these methods is to replace the exact number of people $\geq k$ with an interval which that number corresponds to. To the best of the author's knowledge, this approach has not been applied in the area of anonymization before.

For anonymizing the demographic data, a new method was developed, which minimizes the information loss given by an information theoretic measure, while still fulfilling the constraint of k-anonymization. To the best of my knowledge, the approach of defining information loss as it is done in this thesis, have not been done before. By taking advantage of some of the properties of this definition, it is shown that the information loss can be computed in a reasonable time, even for a high number of dimensions (the computation mainly depends on the number of leaves, see equation (29)). The new method creates, so called, strict multidimensional partitionings and minimizes the informations loss by using a genetic algorithm. It is also shown that this algorithm can find better solutions, in a reasonable time, compared to a previously presented greedy algorithm. Also, this algorithm is not strongly tied specifically to the full anonymization method. Hence, it can separately be used to anonymize any kind of demographic data and does not depend on the other methods described in this thesis.

References

[1] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *In ICDE*, pages 376–385. IEEE, 2008.

[2] Roberto J. Bayardo. Data privacy through optimal k-anonymization. In *In ICDE*, pages 217–228, 2005.

[3] Elisa Bertino. Secure anonymization for incremental datasets. 4165/2006:48–63, 2006.

[4] Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 279–288, New York, NY, USA, 2002. ACM.

[5] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 49–60, New York, NY, USA, 2005. ACM.

[6] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 25–, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Maurizio Atzori Mehmet Ercan Nergiz and Yucel Saygin. Perturbation driven anonymization of trajectories. In *Technical report, CNR*, 2007. Submitted for conference publication.

[8] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In Alin Deutsch, editor, *PODS*, pages 223–228. ACM, 2004.

[9] Anna Monreale, Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Dino Pedreschi, Salvatore Rinzivillo, and Stefan Wrobel. Movement data anonymity through generalization. *Trans. Data Privacy*, 3:91–121, August 2010.

[10] C.L. Blake D.J. Newman and C.J. Merz. UCI repository of machine learning databases, 1998.

[11] Ruggero G Pensa, Anna Monreale, Fabio Pinelli, and Dino Pedreschi. *Pattern-preserving k-anonymization of sequences and its application to mobility data mining*, volume 397, page 44. Citeseer, 2008.

[12] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13:1010–1027, 2001.

[13] Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, October 2002.

[14] Ke Wang and Benjamin C. M. Fung. Anonymizing sequential releases. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 414–423, New York, NY, USA, 2006. ACM.

[15] Xiaokui Xiao and Yufei Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 689–700, New York, NY, USA, 2007. ACM.

[16] Roman Yarovoy, Francesco Bonchi, Laks V. S. Lakshmanan, and Wendy Hui Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 72–83, New York, NY, USA, 2009. ACM.