Data Anonymization Method — An Analysis

William Vedin John Lindström
Email: {wilve383, johli305}@student.liu.se
Supervisor: Jenni Reuben, jenni.reuben@foi.se
Project Report for Information Security Course (TDDD17)
Linköpings universitetet, Sweden

Abstract

The objective of data anonymization is to ensure that individual entries in a dataset cannot be identified in the supposedly anonymous dataset while still maintaining the utility of the data. One of the most common methods for this is k-anonymity. In order to study this method, three experiments were conducted. The first experiment was conducted using the ARX tool to observe the impact of applying the same anonymization methods on three datasets of different sizes. The second experiment was conducted using the ARX tool to observe the impact of changing a specific quasi identifier had on data loss when combined with additional quasi identifiers. The third experiment was conducted by attempting to develop a Python implementation to ensure k-anonymity of a dataset.

The first experiment showed a trend of a smaller amount of data lost through anonymization as the size of a dataset increased. It also showed a trend of a proportionally larger amount of the data loss was the result of generalization compared to suppression as the size of a dataset increased.

The second experiment concluded that the impact of exchanging a singular quasi identifier from one with relatively few unique records to a quasi identifier with a larger number of unique records increased the amount of data loss through anonymization with one additional quasi identifier. The proportion of data loss caused by changing between the quasi identifier with a lower domain value to the quasi identifier with a higher domain value was lower when combined with five auxiliary quasi identifiers.

For the third experiment, the implementation was ultimately unsuccessful but still concluded that it is difficult to test a program that has the purpose of verifying that something else is working correctly. It also concluded that because querying a dataset for- and altering a single record is an expensive operation, it is necessary to keep this to a minimum in order to achieve a reasonable execution time.

1. Introduction

A big problem when it comes to working with datasets from the real world is to ensure the privacy of all the individual entries in the dataset. This is usually done by anonymising the dataset by replacing some of the information that might be used to identify the individuals with synthetic data. This project aims to conduct three experiments to find some interesting properties about the k-anonymity privacy model and algorithm.

The experiments conducted for this project are as follows.

1. Generalization & data loss for differently sized datasets
2. Impact of number of unique quasi identifier values
3. Implementing a Python program to achieve k-anonymity

2. Background

This chapter aims to provide the necessary background information to understand the content of this report.

2.1 Identifiers

There are four different identifiers used in this report: explicit identifiers, quasi identifiers, sensitive attributes and non-sensitive attributes.[1]

Explicit identifiers (often referred to as just *identifiers*) are attributes that can explicitly identify an entry in a dataset, such as a name or a unique medical ID.

Quasi identifiers are a set of non-sensitive attributes that when combined may lead to identification of a supposedly anonymous entry in the database.

Sensitive attributes (sometimes *confidential outcome attribute*s) are attributes that contain sensitive information about the entry, for example salary or a disease.

Non-sensitive attributes are the remaining attributes, meaning the attributes that are unlikely to be useful for deanonymizing a user, or attributes that otherwise are not sensitive to the user.[2]

2.2 k-anonymity

K-anonymity is a privacy model that ensures that entries in a dataset are indistinguishable from k-1 other entries. In order for a dataset to satisfy k-anonymity, every record in the dataset has to share the values of its quasi identifiers values with at least k-1 other records.[3] In order to achieve this, either suppression or the generalization is used. Suppression is a method of removing attributes from the dataset completely, whereas generalization is turning specific attributes into more general ones. A common way of using generalization is to use age spans rather than discrete values for data entries. Below are examples illustrating this.

Name	Zip code	Age
Erik	53436	25
Rolf	53436	23
Jens	53435	30
Theo	53435	35

Table 1: 0-anonymous dataset

In table 1, name is obviously an identifying attribute, and is a clear way to link a dataset entry to a person. In order to attain any form of anonymity, this cannot be included and as such this attribute will be suppressed from the dataset completely.
This is not enough however, as the age values of each entry in the dataset are different. By generalizing the age attribute and suppressing the name field, the following table can be produced.

[1] https://link.springer.com/article/10.1007/s10618-005-0007-5

[2] https://arx.deidentifier.org/overview/privacy-criteria/

[3] https://cloud.google.com/dlp/docs/compute-k-anonymity

Zip code	Age
53436	20-29
53436	20-29
53435	30-39
53435	30-39

Table 2: 2-anonymous dataset

In table 2, the pair of rows 2 and 3 as well as 4 and 5 are indistinguishable from each other. Each entry has at least 2-1 seemingly identical fields for the quasi identifiers zip code and age, and as such the anonymized dataset satisfies 2-anonymity.

2.3 l-diversity

L-diversity is a privacy model used in conjunction with k-anonymity. In order for a dataset T to satisfy l-diversity, it implies that for every set of records with the same value for each quasi identifier, there are at least l distinct values for each of the sensitive attributes of a dataset. In contrast to k-anonymity that implies a certain indistinguishability between values of quasi-identifiers in a dataset, l-diversity implies indistinguishability between sensitive attributes in a dataset.

Assume the dataset from table 2 is a log of medical entries, with the added attribute of the diagnostic from the medical visit. This would not be an identifying attribute as it on its own would not link the entry to a person. In this case it would not be a quasi identifier as it would not assist in identifying a sole user from the dataset, as it is a sensitive attribute.

Zip code	Age	Diagnosis
53436	25	Cold
53436	23	Flu
53435	35	Tetanus
53435	35	Tetanus

Table 3: 0-diverse dataset

In table 3, although the third entry is indistinguishable from the fourth, one can deduct that by knowing that an individual person had been admitted to the medical facility, living in the zip code 53435, and is of age of 35 had been diagnosed with tetanus. This implies that table 3 does not achieve 2-anonymity in the regard of l-diversity.

L-diversity uses the same suppression and generalization methods used in k-anonymity to achieve a higher degree of anonymity. Applying generalization and suppression to the dataset in table 3 results in the following table.

Zip code	Age	Diagnosis
5343*	20-39	Cold
5343*	20-39	Flu
5343*	20-39	Tetanus
5343*	20-39	Tetanus

Table 4: 4-diverse dataset

In table 4, part of the zip code is generalized, and the precision of the age attribute is reduced as the span is increased. This results in the sensitive diagnostic attribute being protected. As the separate users cannot be mapped to any of the sensitive attributes, the dataset in table 4 has the l-diversity property of 4-diversity. It should also be noted that the dataset now has the property of 4-anonymity.

2.4 δ-presence

Delta-presence is a privacy model that determines the highest probability of an individual record in a bigger population is an entry in the smaller analyzed dataset.[4] Unlike l-anonymity and k-anonymity, the δ parameter in δ-presence is not a discrete integer, but rather a value in the interval [0, 1]. A low δ parameter is also considered more secure, unlike k-anonymity and l-diversity where a high parameter value is prefered.
1-presence implies a 100% probability of a user from the larger population being part of the smaller dataset, and 0-presence in contrast implies a 0% probability.

Assume the scenario of a lottery that chooses to publicly display data including ZIP code and the age of their winners as shown in table 5.

Zip code	Age
53436	20-29
53436	20-29
53436	30-39
53436	30-39

Table 5: 0.4-presence dataset

Public statistics show that there are ten people in the zip code in the age interval [20, 29] and five people in the age interval [30, 39]. For the younger age group, the probability of a member of the larger population is 2/10 = 0.2. The probability for a member of the older age group is 2/5 = 0.4. As the δ-presence is defined by the greatest ratio in the dataset, the dataset in table 5 has a 0.4-property.

Like k-anonymity and l-diversity, the methods of suppression and generalization can be applied to attain a lower δ-value for the produced dataset, as shown below in table 6.

Zip code	Age
5343*	20-29
5343*	20-29
5343*	30-39
5343*	30-39

Table 6: 0.04-presence dataset

For simplicity, assuming that the population in the nine other ZIP codes starting with 5343 have ten and five people in the spans [20, 29] and [30, 39] respectively, this would give the respective probability of 2/100=0.02 and 2/50 = 0.04. This implies that the dataset in table 6 has the property of 0.04 presence.

2.5 Data loss

When generalizing a dataset, some of the data will inherently be lost. The goal of data generalization is usually to achieve the desired level of anonymity, while minimizing the amount of data lost in the process. Data loss can be measured using different metrics for different types of columns[5]. In this project, the metric used to measure data loss is discussed in section 2.6.

2.6 ARX Data Anonymization Tool

In order to both implement and measure the generalization strategy, the tool *ARX Data Anonymization,* hereafter referred to as ARX, is used. ARX allows the user to load datasets and define an anonymization strategy to be automatically applied to the dataset. It is possible to set rules for how the program will automatically generalize each quasi identifier. This can be done in multiple generalization

[4] https://cloud.google.com/dlp/docs/compute-d-presence

[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5504813/

levels per quasi identifier, as specified by the generalization hierarchy set for each quasi identifier. An example of a generalization of the quasi identifier date of birth is shown in table 6 below.

Level 1	Level 2	Level 3
1987/10/2	1987/10	1987
1987/10/5	1987/10	1987
1987/1/2	1987/1	1987
1987/6/6	1987/6	1987

Table 6: Example of generalization levels

After the anonymization strategy has been entered into the program, the user can choose between different anonymization models and their corresponding parameter values to ensure the anonymity of the dataset. In this project, k-anonymity has been used for this purpose. If the user selects k-anonymity and a k-value and runs the anonymization strategy, it will be presented with all of the solutions that achieve k-anonymity. These solutions will be in the form of the level of generalization that has been applied to each quasi identifier that was necessary to achieve k-anonymity. For each solution, the user can then view the amount of data loss, what percentage of rows have been suppressed, and an ARX-generated score for both anonymity and data loss. ARX will also highlight the solution that achieves k-anonymity while losing as little data quality as possible, and what permutations of quasi identifier

Figure 1 is an example of the solutions for the current configuration as presented by ARX. Each node represents a solution based on the transformations that were applied, with each number corresponding to a level in the generalization hierarchy for this quasi identifier. The golden node is the optimal solution, and the 4th integer in this node implies that the 5th generalization specified for that quasi identifier is used in the optimal solution. Although not shown here, ARX shows the user a specific utility measure for each node. Although multiple utility metrics are available, this project uses data loss and suppression level exclusively.

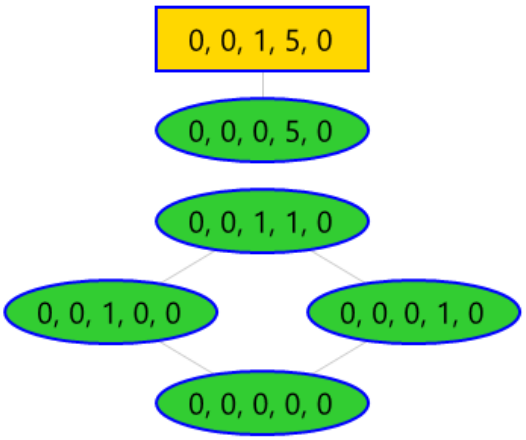


Figure 1. ARX solution space

2.7 Implementing a Python program to verify k-anonymity

As part of this project, an attempt at implementing a basic Python program to verify k-anonymity for a dataset was made. The goal for this program was that it would be able to handle a large dataset in the .csv file format, check for violations of k-anonymity, and finally generalize or suppress sensitive records in the dataset if it did not already satisfy k-anonymity.

The purpose of this program was to create a baseline algorithm for achieving k-anonymity, and measure the output for comparison with the ARX tool.

As a way of handling the large datasets used in this project, an external library was used. The library chosen for this task was *Dask*. Dask describes itself as a *scalable library for parallel computing in Python*. It is a wrapper library that

implements features that make it easier to work with large datasets using functionality from the more well known libraries *Pandas* and *NumPy*. Some of the features offered by Dask are delayed computation, dividing the datasets into chunks to save memory, and parallel computing, all on top of the already existing functionality from Pandas and NumPy.[6]

Dask, through Pandas, offers a large amount of features for handling big datasets. It is possible to read a csv file through the use of a Dask library function, and save it in a Dask *dataframe*, which is a large collection of data that can be partially loaded into memory when needed. These dataframes formed the basis upon which the algorithm was built.

3. Method & experiments

This chapter aims to explain the experiments that were conducted for this project.

3.1 Generalization & data loss for differently sized datasets

This experiment investigates the generalization strategies and the data loss for applying the ARX k-anonymity function on differently sized subsets of a larger dataset. The dataset used for this experiment was the *A&E Synthetic Data*[7], containing 18 columns. This dataset originally contains over 65 million records, and can be difficult to use in its entirety. For this reason, the experiments were conducted using subsets of this dataset. The subsets contained 1000, 10 000 and 1000 000 records respectively. The records in this dataset are independent of each other, and the larger subsets contain the smaller subsets.

For this experiment, ARX was configured to find an optimal solution in regards to data loss that satisfies 5-anonymity. All the solutions in the solution space were rated using ARX internal scoring system for measuring data loss[8], and produced a solution for each permutation of generalization level for each quasi identifier that satisfies 5-anonymity. The optimal solutions for each subset were then compared to find how the size of the subset affected how much generalization and suppression had to be performed on the dataset to satisfy 5-anonymity.

In this experiment, the following columns from the dataset were used as quasi identifiers.

1. Decile from LSOA
2. Age_Band
3. Sex
4. Arrive_Date
5. Arrive_HourOfDay
6. Time_Mins
7. HRG

Quasi identifiers 1, 4 and 6 were attributed 3, 2 and 4 generalization levels respectively. The remaining quasi identifiers were either unsuitable for generalization or were already generalized, and as such were not attributed any generalization hierarchies.

3.2 Impact of unique quasi identifier values

This experiments investigates the impact of changing between for different combinations of quasi identifiers when using ARX to achieve 100-anonymity on a 1000k record subset from the *A&E Synthetic data* dataset.

In this experiment, a combination of quasi identifiers will be set up and their original and transformed values will be examined. These combinations will split this experiment into the four sub experiments below:

[6] https://dask.org/

[7] https://data.england.nhs.uk/dataset/a-e-synthetic-data

[8] https://arx.deidentifier.org/overview/metrics-for-information-loss/

A1 consists of studying a quasi variable assigned the name Q1, which is used with one auxiliary quasi identifier to anonymize the dataset.

A2 consists of studying Q1, which in this sub experiment is used with the auxiliary quasi identifier used in A1 together with four additional quasi identifiers. The number four was chosen as fewer additional identifiers didn't result in any significant change.

B1 consists of studying a quasi variable assigned the name Q2, which is used with the same auxiliary quasi identifier used in A1.

B2 - consists of studying Q2 with the same auxiliary quasi identifiers studied in A2.

The different combinations of quasi identifiers are shown in table 7.

Sub experiment	Quasi Identifiers
A1	Q1, Q3
A2	Q1, Q3, Q4, Q5, Q6, Q7
B1	Q2, Q3
B2	Q2, Q3, Q4, Q5, Q6, Q7

Table 7: Quasi identifiers used in experiment 2

3.3 Implementing a Python program to verify k-anonymity

For this program, the functionality was split in two. One part to check for k-anonymity, and the other part to suppress and generalize data.

Verifying k-anonymity

At first, a naive approach was taken towards implementing the first part as a proof of concept. The algorithm used was as written below.

Input: value of k, list of quasi identifiers, name of csv-file containing the dataset.

1. Read the csv-file to a dataframe.
2. Find all unique values in the columns marked as quasi identifiers.
3. Generate all ordered permutations for the unique values in step 2. These are the equivalence classes for the dataset.
4. Query the dataframe for each equivalence class. Make sure that the amount of records in each equivalence class is either zero, so that it does not exist in the dataset, or larger than k-1. If none of these are true, mark it for generalization or suppression.

There are some obvious problems with this naive approach. Step 3 above will generate every possible equivalence class. For quasi identifiers with a large domain, this could generate thousands or millions of equivalence classes that are not in the dataset, and checking these is obviously not a good use of resources. Querying a dataset in Dask is a slow operation, so this algorithm was not feasible for any dataset large enough to be anything other than a proof of concept, as it simply took too long to run.

After this initial approach was scrapped, an alternative method was used. After discussion with our supervisor, the following algorithm was devised.

Input: value of k, list of quasi identifiers, name of csv-file.

1. Read csv-file to a dataframe.
2. Group the data records by the quasi identifiers.
3. Check all the groups generated. If any groups have fewer than k-1 records,

mark them for suppression and generalization.
4. For each group, get the column name and corresponding values. Query the dataframe for a list of columns with their corresponding values to get the records in each group.
5. Mark these records for generalization and suppression.

Implementing generalization and suppression
After the records had been marked for generalization and suppression, they were sent as input to a generalization- and suppression algorithm. The idea behind this algorithm was to implement it as a higher-order function, taking the records to be altered as well as a function to apply to them as arguments to the function.

4. Results

In this section , the results from the performed experiments are presented.

4.1 Generalization & data loss for differently sized datasets

In this subsection, the results from experiment one are presented. For each of the three dataset, this includes the reduced solution space, the number of records, the optimal solution and its data loss.

All solution space figures are presented in a reduced form with some nodes hidden to increase readability. Their full forms can be found in Appendix A.

1k data

In figure 2 and table 8 below, the data from the experiment with the subset of 1k records is presented. The optimal solution in this case is the solution with the largest possible amount of generalization applied to each quasi identifier.

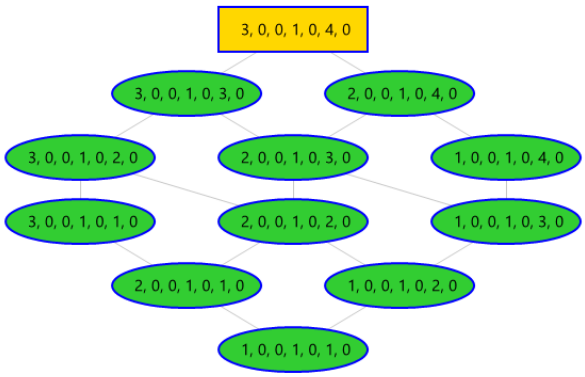


Figure 2: Reduced solution space for 1k rows

Property	Value
Number of records	1000
Optimal solution	[3, 0, 0, 1, 0, 4, 0]
Data loss in optimal solution	0.9179
Suppressed records	88.01%

Table 8: Metrics from the dataset with 1k records

10k data

In figure 3 and table 9 below, the data from the experiment with the subset of 10k records is presented. The optimal solution in this case is the solution with the largest possible generalization for two of the three quasi identifiers with multiple generalization levels. For the last quasi identifier, the second highest possible generalization is applied.

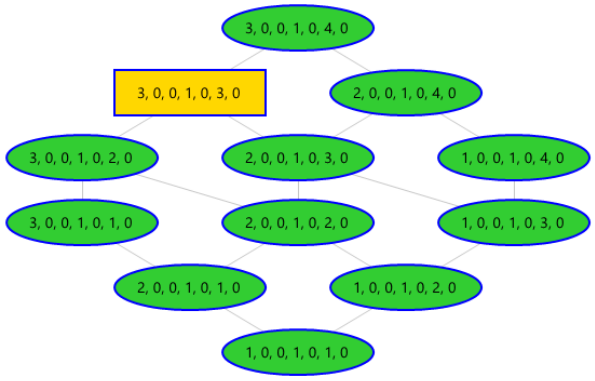


Figure 3: Reduced solution space for 10k rows

Property	Value
Number of records	10 000
Optimal solution	[3, 0, 0, 1, 0, 3, 0]
Data loss in optimal solution	0.3879
Suppressed records	25.336%

Table 9: Metrics from the dataset with 10k records

1000k data

In figure 4 and table 10 below, the data from the experiment with the subset of 1000k records is presented. In this optimal solution, the first and third variable quasi identifiers are two levels below their highest generalization level, and the second quasi identifier is at its highest generalization level.

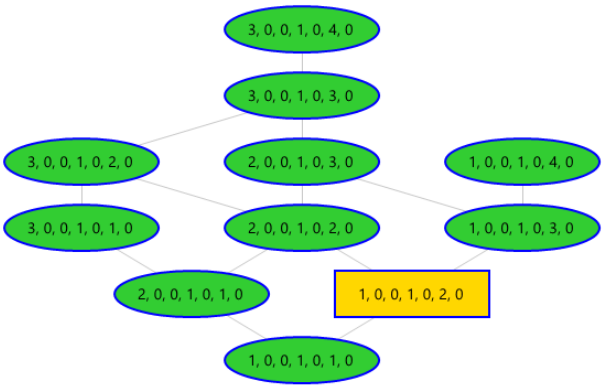


Figure 4 reduced solution space for 1000k dataset

Property	Value
Number of records	1000 000
Optimal solution	[1, 0, 0, 1, 0, 2, 0]
Data loss in optimal solution	0.0793
Suppressed records	1.204%

Table 10: Metrics from the dataset with 1000k records

Overview of the experiment

Subset size	ARX Loss Metric	Suppression rate	$\frac{Supression\ Rate}{ARX\ Loss\ Metric}$
1k	91.79%	88.01%	95.88%
10k	38.79%	25.34%	65.33%
1000k	7.94%	01.20%	15.11%

Table 11: Metrics for the optimal solution for each dataset

Table 11 contains the ARX metrics for the optimal solution for each of the subsets. The suppression rate divided by the ARX loss column is a relative metric showing the amount of records that had to be completely suppressed in order to satisfy k-anonymity.

4.2 Impact of unique quasi identifier values

In this subsection, the results from experiment two are presented. For each of the four sub experiments, this includes the number of quasi identifiers, unique records for the quasi identifier being observed and the number of equivalence classes prior to the transformation. This also includes the data loss, generalization level and number of equivalence classes for the optimal solution produced.

A1

In figure 5 and table 12, the data from the A1 sub experiment is presented. It is worth noting

that Q1 is the first element in the solution space matrices. In the optimal solution, Q1 is only generalized to the first level. By looking at table 12, this information can be used to conclude that there are 49 unique entries in Q1 in this solution. This is in comparison to the 1461 unique entries in Q1 prior to the transformation.

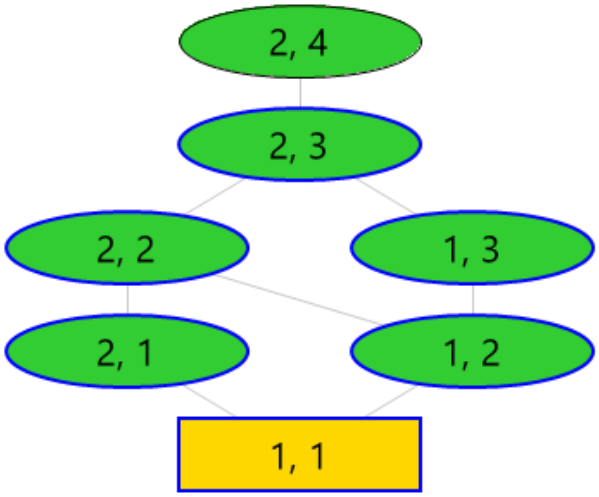


Figure 5: Reduced solution space for A1

Property	Value
Number of Quasi identifiers	2
Unique records in Q1	1461
Unique records in generalized Q1	[1461 ,49, 5]
Number of EC prior	92545
Number of EC after in optimal solution	343
Data loss in optimal solution	5.79%
Generalization level of Q1 in optimal solution	1
Suppressed records	0%

Table 12: Data for A1

A2

In figure 6 and table 13 below, the data from the A2 sub experiment is presented. Q1 is the 3rd element in the solution space matrices. This means that in the optional solution, Q1 has 5 unique entries, as opposed to the 1461 prior to transformation.

Property	Value
Number of Quasi identifiers	6
Unique records in Q1	1461
Unique records in generalized Q1	[1461 ,49, 5]
Number of EC prior	826694
Number of EC after in optimal solution	955
Data loss in optimal solution	10.909%
Generalization level of Q1 in optimal solution	2
Suppressed records	3.82%

Table 13: Data for A2

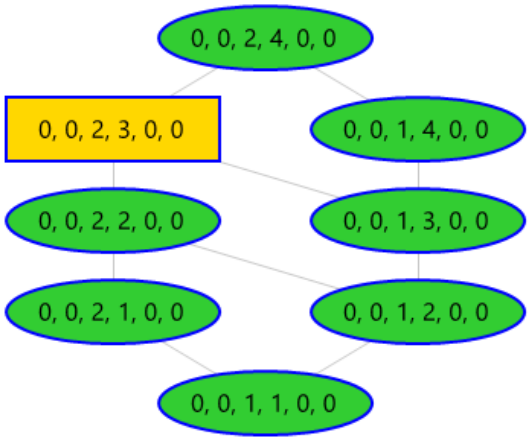


Figure 6: Reduced solution space for A2

B1

In figure 7 and table 14 below, the data from the B1 sub experiment is presented. Q2 is the first element in the solution space matrices. This means that Q2 has 11 unique records in the optimal solution, as it had prior to the transformation.

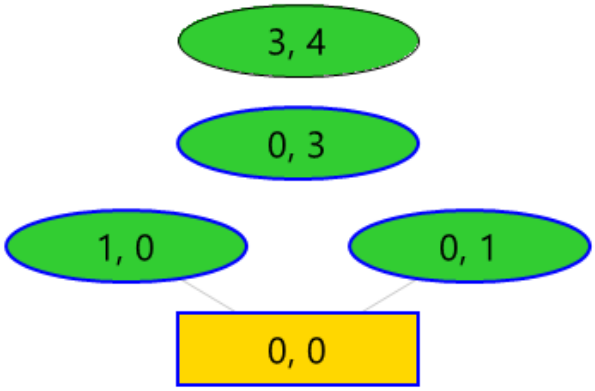


Figure 7: Reduced solution space for B1

Property	Value
Number of Quasi identifiers	2
Unique records in Q2	11
Unique records in generalized Q2	[11, 6, 2]
Number of EC prior	1497
Number of EC after in optimal solution	556
Data loss in optimal solution	2.38%
Generalization level of Q2 in optimal solution	0
Suppressed records	2.38%

Table 14: Data for B1

B2

In figure 8 and table 15, the data from the B2 sub experiment is presented. Q2 is the first element in the solution matrices. This means that in the optimal solution, Q2 has 6 unique entries, as opposed to 11 prior to the transformation

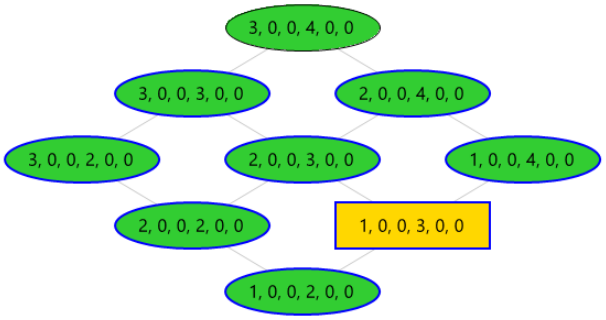


Figure 8: Reduced solution space for B2

Property	Value
Number of Quasi identifiers	6
Unique records in Q2	11
Unique records in generalized Q2	[11, 6, 2]
Number of EC prior	91195
Number of EC after in optimal solution	998
Data loss in optimal solution	8.96%
Generalization level of Q2 in optimal solution	1
Suppressed records	3.78%

Table 15: Data for B2

Overview of the experiment 2

Property	Value
Data loss A1 / Data loss B1	2.43
Data loss A2 / Data loss B2	1.22
Data loss A2 / Data loss A1	1.88
Data loss B2 / Data loss B1	3.76

Table 16: Data loss metrics for experiment 2

Table 16 contains the data loss metrics from A1, A2, B1 and B2. The ratio in row shows that with one additional quasi identifier, the Q1 quasi identifier data loss is 2.43 times higher compared to when using Q2. Row 2 shows that the corresponding factor when using five

additional quasi identifiers is only 1.22. Row 3 shows that the data loss increase from going from 2 to 6 quasi identifiers including Q1 increases dataloss by a factor of 1.88. In Row 4, it shows that the corresponding magnitude when using Q2 is 3.76.

4.3 Implementing a Python program to verify k-anonymity

In this subsection, the results from experiment 3 are presented.

Verifying k-anonymity
An attempt was made at implementing both the k-anonymity algorithms presented in chapter 3.3. None of them were successful for different reasons. The first algorithm was very time inefficient, and not viable for any real world applications. It is also not certain that it worked, as it was never tested due to time- and computation limitations.

The second algorithm was also implemented. Unlike the first algorithm, this algorithm was able to terminate within a reasonable timeframe. The problem with this algorithm was that it was unable to be tested, due to problems with the generalization- and suppression algorithm. Because of this, development of the second algorithm for verifying k-anonymity had to be dropped due to time limitations.

Implementing generalization and suppression
At the start of the project, the idea for the generalization- and suppression algorithm was to create a crude version that suppressed every value in a sensitive column for the entire dataset. This was implemented through a higher-order function and a lambda function that simply replaced all values in a column with a chosen value. This worked, but it was not very practical or compatible with the k-anonymity algorithms, which would have worked by finding sensitive records. It also caused massive data loss, so this idea was quickly abandoned.

The idea behind the second version of this algorithm was for it to query the dataset for the sensitive records and alter the values of the quasi identifiers. This proved to be a difficult task, as Dask does not provide a simple way of editing a single cell in a dataframe object. A large amount of time was spent trying to find a time efficient way of doing this, however no practical way was found. For this reason, the experiment had to be abandoned and this seemingly trivial operation proved to be the single most important point of failure in the experiment.

5. Analysis

This chapter aims to analyze the different experiments that have been conducted for this project.

5.1 Generalization & data loss for differently sized datasets

The idea of experiment 1 was to observe to which degree the different datasets were generalized. This can be interpreted in many ways. One way to interpret this is to observe a trend that points toward how as the dataset increases, the level of which the quasi identifiers are generalized decreases.

At first glance, an assumption one might make regarding this is that as the number of the dataset increases, the ratio of data loss from generalization and suppression would be skewed towards more data being lost to suppression rather than generalization, as generalization as previously mentioned decreased when the dataset was increased in size.

This assumption is however not true, as one can see in table 11 where this ratio is calculated. This metric shows that for the 1k subset, 95.88% of the data from the initial dataset is lost due to suppressed values. This means that only the remaining 4.12% of the initial data loss is from generalization, as of course the total data loss is caused by the amount of data lost from suppression in addition to generalization. As the size of the dataset increases, suppression accounts for only 65.33% and 15.11% of data loss for the 10k and 1000k datasets respectively.

These observations instead imply that although the level to which the quasi identifiers have to be generalized as the number of records in a dataset increases, the proportion of data loss caused by generalization also increases. Intuitively these findings make sense when in the case, like in this experiment, the k-parameter of k-anonymity stays constant as the data set records increase. For a smaller data set, there are simply too few records that are close enough to other records to be generalized to be indistinguishable. For these smaller datasets, these outliers instead have to be suppressed completely.

For a larger dataset however, there are more records which can be made indistinguishable to other records through generalization, resulting in fewer outliers having to be suppressed in order to maintain the property of k-anonymity of the produced dataset. In this case, as more records are generalized, more of the remaining entries will be less accurate than they were before the generalization, and as such a larger proportion of the total data loss will be caused by generalization.

5.2 Impact of unique quasi identifier values

The idea of this experiment was to observe the impact of two different quasi identifiers, first in conjunction with one auxiliary quasi identifier, and again with five auxiliary quasi identifiers. The quasi identifier Q1 contains significantly more unique values than Q2. This makes the anonymization process more complicated, as this implies that there will be fewer entries that are indistinguishable from each other.

In A1, one can observe that no suppression has to be made. This is partly due to the fact that the dataset has a wide array of unique entries that are fairly evenly distributed. With an even distribution, this large amount of unique values can be put into a higher amount of smaller spans compared to what an identifier with a smaller amount of unique entries would without losing as much data in the process. This can be observed in the fact that Q1 is only generalized once in the optimal solution in A1, which has a total of 49 unique values.

Contrary to this, a quasi identifier with very few identical values cannot be generalized in many ways without suffering a bigger loss. This can be intuitively explained by assuming a dataset with 1000 unique entries for a quasi identifier. If this identifier is generalized once into 100 new generalized values, some data is lost. Generalizing these values into 10 broader groups will make them more indistinguishable, but once again most likely result in more data lost. This is the case of experiment B1. For this experiment, generalizing Q1 will result in too much data loss, so the optimal solution is obtained by suppressing the values that prevent the dataset from maintaining 100-anonymity.

Another interesting observation to make from this experiment is by looking at the data loss caused by the switching of quasi identifiers Q1

and Q2 when multiple auxiliary quasi identifiers are added rather than just the one in A1 and B1. By comparing the loss from A1 and B1, the loss in A1 is 2.43 greater than the data loss in B1. However in A2 and B2, this factor is only 1.22. This data can together show that although the impact of a quasi identifier that is hard to anonymize can be fairly substantial when only a few quasi identifiers are used, its impact will be comparatively smaller as additional quasi identifiers are added.

5.3 Implementing a Python program to verify k-anonymity

Difficulties and why the implementation failed

In the end, the attempts at creating an algorithm to achieve k-anonymity for a dataset were unsuccessful. This was at least partly because a suppression algorithm could not be completed within the timeframe for this project. Instead of using the results from this algorithm as was originally planned, this analysis will instead focus on how it could be implemented and how it could be tested.

While Dask is a generally flexible and very useful framework, it was likely not the best solution for this implementation. It is not a very mature framework, so a lot of extra work had to be put in to work around missing functionality. In the end, it was the seemingly trivial task of changing a single cell in the dataframe that proved to be the final nail in the coffin for this implementation. This is a feature that exists in Dask according to its documentation, however for some reason the developers were not able to solve the error that occured when trying to do so.

Alternative solutions

An alternative approach to this implementation would have been to use Python library *Sqlite3*. Sqlite3 is a native library for integrated database support, and is optimized for Python. While a database would be a less flexible solution than Dask, Sqlite3 is a more mature library and carries all features that a SQLite database does. It is also possible to read a csv-file and save it as a database, so not much would change in that aspect. It is also very fast, so execution time is unlikely to be affected negatively by using it over Dask.

Testing a functional solution

Testing an algorithm where a part of the task is to test something else, in this case testing for k-anonymity, is very difficult, as if the algorithm has been constructed incorrectly it is likely that the algorithm does not work but will still terminate correctly. It is then difficult to say whether the algorithm worked or not.

If a working k-anonymity verifier and anonymizer had been achieved, it would be possible to manually set up different scenarios where the tester knows beforehand what operations the algorithm should make on a carefully constructed test data set. The tester can then observe and compare the resulting dataset with the expected output and see if the algorithm has been implemented correctly. The problem with this approach is that it would be very time consuming to construct these tests, and it would be highly impractical or impossible to gain confidence that it works for large datasets in this way.

The obvious and easiest way of testing a functional algorithm would be to run its output through another program that is already known to work, such as ARX. This has the obvious drawback of being dependent on another tool that does the same thing as the tool that is being implemented, and is as such not considered to

be a good solution in the general case. If the purpose of developing an implementation is to implement custom, additional metrics of any kind, this could be a useful way of testing it.

6. Conclusions

In this chapter, the conclusions for each experiment will be stated.

6.1 Generalization & data loss for differently sized datasets

When comparing three datasets of different sizes with the same set of quasi identifiers and anonymized to obtain the same 5-anonymity, the trend shows that the bigger the data set is, the smaller the proportional data loss is. As the data set increased, it was also observed that a proportionally lower amount of data loss was the result of suppressed values. This also implied that as a dataset increased in size, a proportionally higher amount of data loss was the result of generalization.

6.2 Impact of unique quasi identifier values

When comparing two different quasi identifiers and their impact along with an auxiliary quasi identifier, it was observed that a quasi identifier with significantly fewer unique values could obtain 100-anonymity with much less data loss than its counterpart with more unique values. When comparing the same two quasi identifiers with five auxiliary quasi identifiers, the proportional difference in data loss was substantially lower between the two.

6.3 Implementing a Python program to verify k-anonymity

While the implementation of such an algorithm for this project was not successful, some conclusions can still be drawn from this experiment.

First of all, k-anonymity is not a simple topic, and testing an implementation of an anonymization tool is not a simple matter. It is possible that the implementation of the k-anonymity for this project was correct, but it was very difficult to test it due to the dysfunctional generalization- and suppression algorithm, as that meant there was no output from the verification program.

It can also be noted that because querying and changing single rows or cells in large datasets is an expensive operation, any algorithm to ensure k-anonymity is likely to be slow unless carefully constructed and optimized. It is necessary to reduce the amount of single row queries and alters as much as possible in order for the algorithm to run as quickly as possible.

References

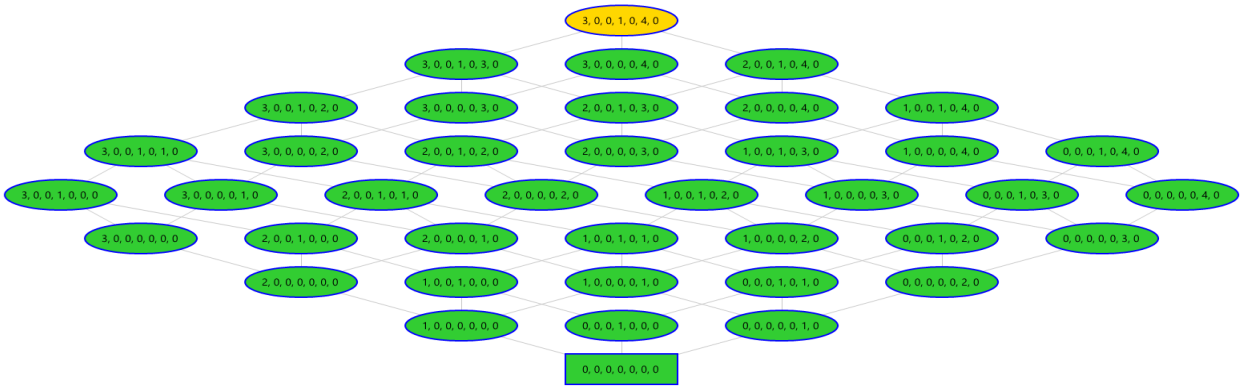
1. Josep Domingo Ferrer, 2005, Ordinal, Continuous and Heterogeneous k-Anonymity Through Microaggregation, accessed 5/11/2022 https://link.springer.com/article/10.1007/s10618-005-0007-5
2. ARX - Data anonymization tool - Privacy models, accessed 5/11/2022 https://arx.deidentifier.org/overview/privacy-criteria/
3. Google cloud - Computing k-anonymity for a dataset, accessed 5/11/2022 https://cloud.google.com/dlp/docs/compute-k-anonymity
4. Google cloud - Computing d-presence for a dataset, accessed 5/11/2022 https://cloud.google.com/dlp/docs/compute-d-presence

5. Hyukki Lee et al., Utility-preserving anonymization for health data publishing, accessed 5/11/2022 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5504813/
6. Dask.org, accessed 5/11/2022 https://dask.org/
7. Data Catalogue England, accessed 5/11/2022 https://data.england.nhs.uk/dataset/a-e-synthetic-data
8. ARX - Data anonymization tool - Data quality models,, accessed 5/11/2022 https://arx.deidentifier.org/overview/metrics-for-information-loss

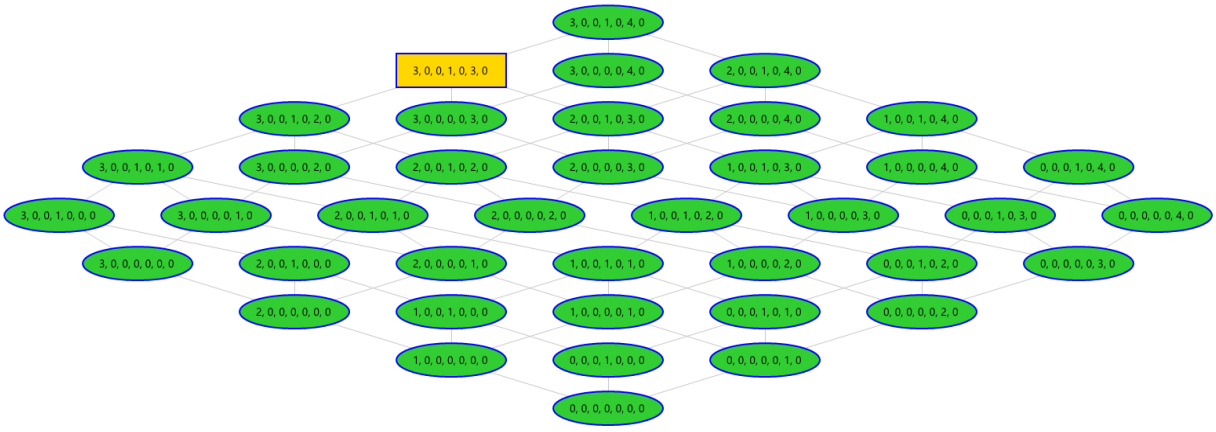
Appendix A

In this appendix, the full solution spaces from experiment 1 and 2 are shown.

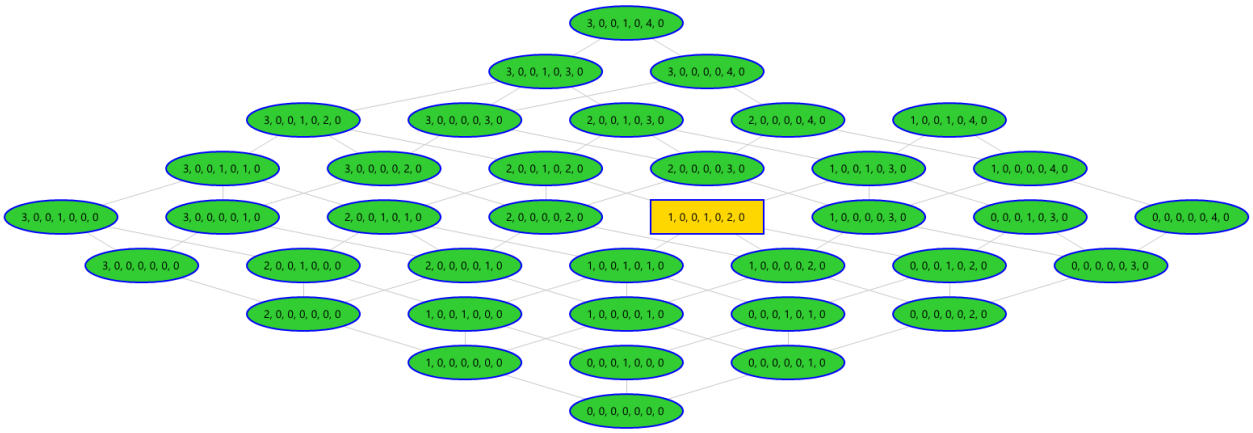
Experiment 1



Complete transformation space of the 1k dataset in experiment 1

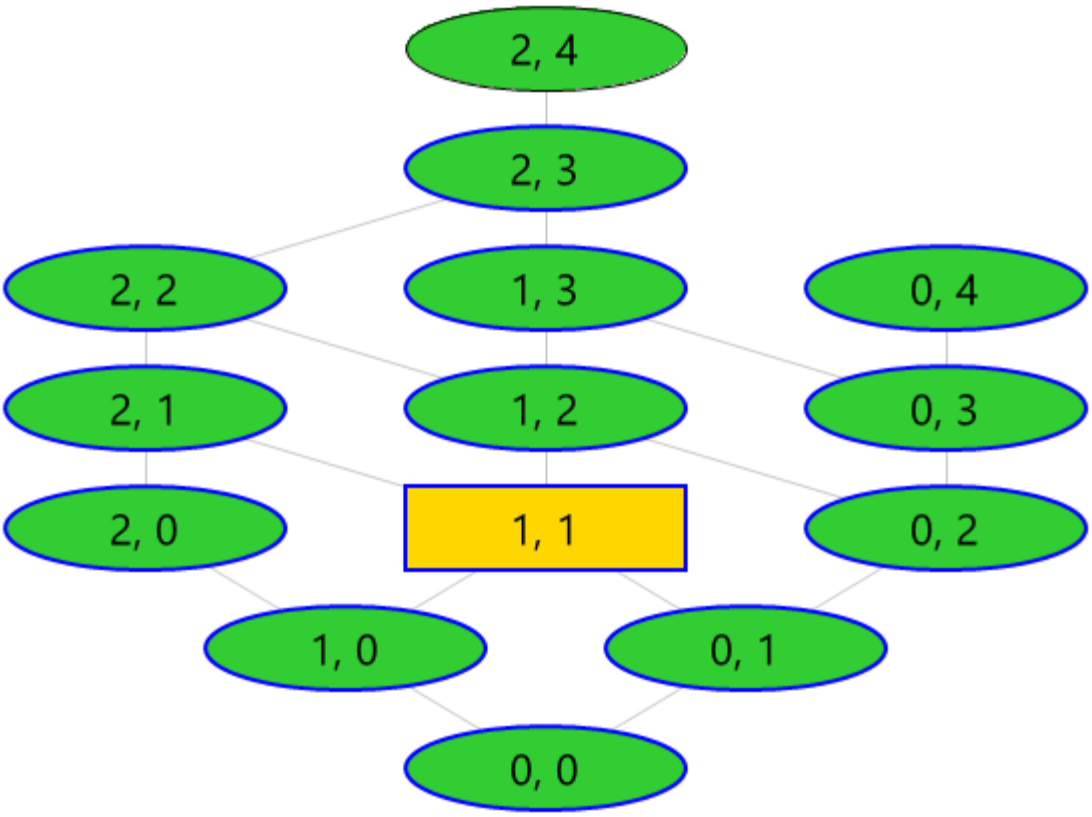


Complete transformation space of the 10k dataset in experiment 1

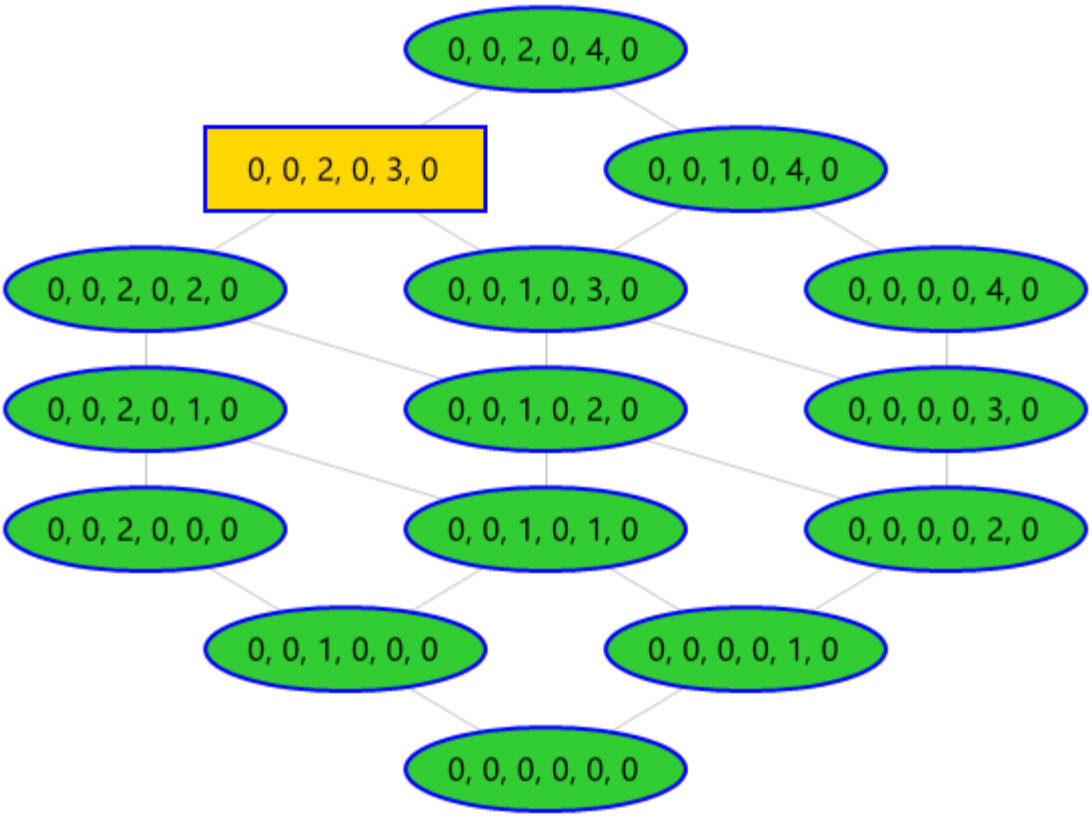


Complete transformation space of the 1000k dataset in experiment 1

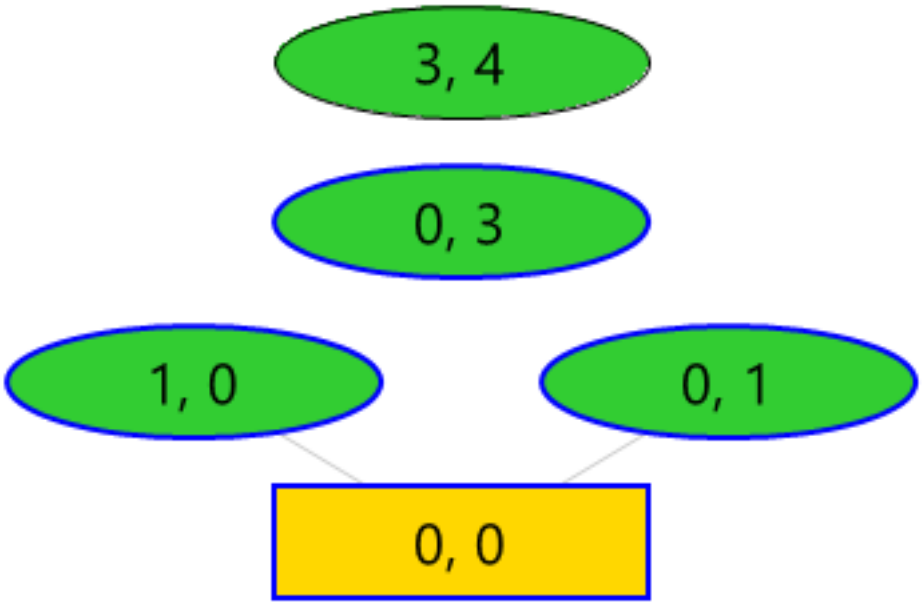
Experiment 2



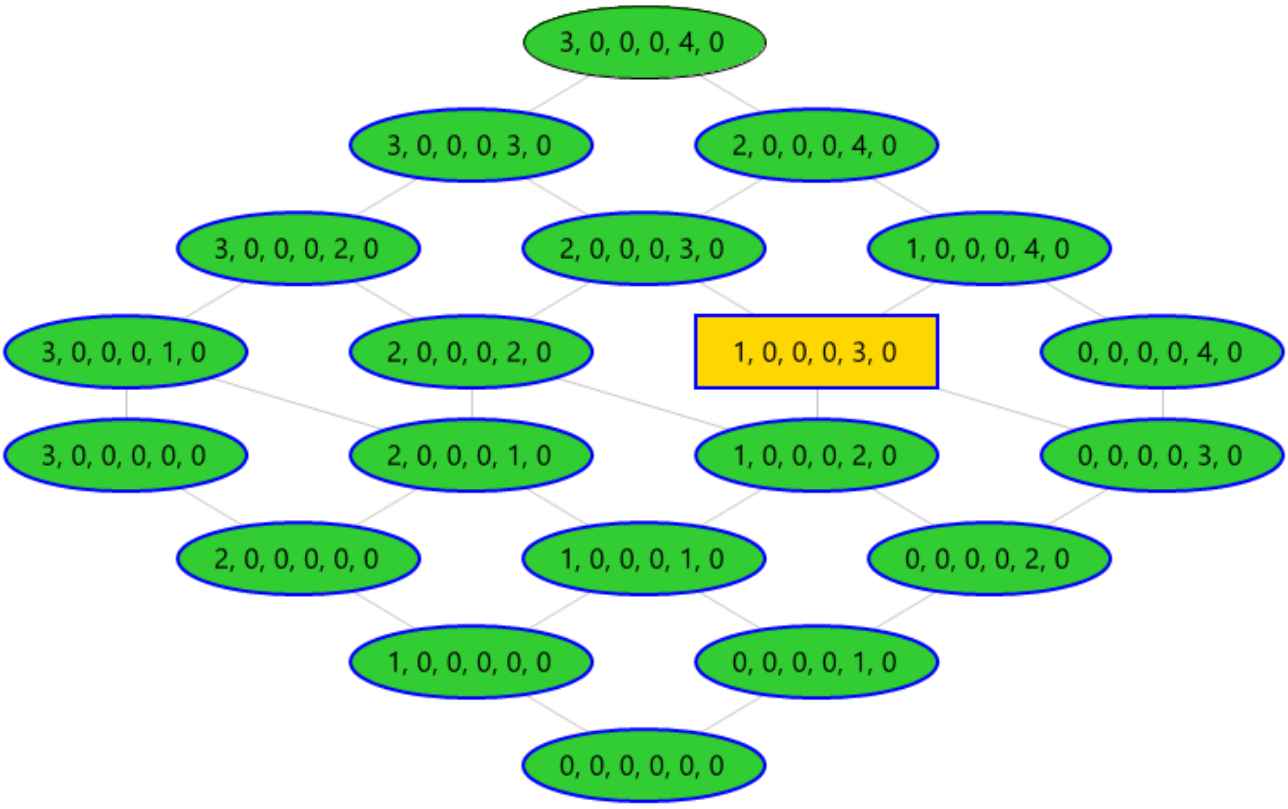
Complete transformation space of A1 in experiment 2



Complete transformation space of A2 in experiment 2



Complete transformation space of B1 in experiment 2



Complete transformation space of B2 in experiment 2