Computational efficiency of k-anonymization incorporating clustering

---

Netshiunda Fhulufhelo Emmanuel

Supervisor:
Prof. Turgay Celik (University of the Witwatersrand)

A research report submitted in partial fulfillment of the requirements for the degree of Master of Science in the field of e-Science

in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

23 May 2020

Declaration

I, Netshiunda Fhulufhelo Emmanuel, declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the field of e-Science at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.

Netshiunda Fhulufhelo Emmanuel

23 May 2020

Abstract

Data publicizing pose a threat of disclosing data subjects associating them to their personal sensitive information. *k*-anonymization is a practical method used to anonymize datasets to be made publicly available. The *k*-anonymization hides identities of data subjects by ensuring that every record of a publicized dataset has at least $k-1$ (k being a natural number) other records similar to it with respect to a set of attributes called *quasi-identifiers*. To minimize information loss, a clustering technique is often used to group similar records before *k*-anonymization is applied. Processing both the clustering and the *k*-anonymization using current algorithms is computationally expensive. It is within this framework that this research focuses on parallel implementation of the *k*-anonymization algorithm which incorporates clustering to achieve time effective computations.

Acknowledgements

I would like to

- Thank God for giving me strength to construct this research report.
- Sincerely thank my supervisor Prof. Turgay Celik, for his help, motivation and guidance in preparing this report, he has always been available when I needed his assistance.
- Sincerely thank my parents and my uncles for the support they have given me.
- Thank Ms Casey Sparkes and everyone involved directly and indirectly in this research.

Sponsors I would sincerely like to thank DST-CSIR National e-Science Postgraduate Teaching and Training Platform (NEPTTP), for funding this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This section presents the background, problem statement, research aims, objectives and limitations of this research.

1.1 Background

Due to recent developments of large quantity of data being collected and stored each day by private and public organizations as well as the realization of the value of data, there arises a demand for data sharing from one organization to another. Analyzing shared data can help to reveal patterns and trends for these organizations gain. For instance, organizations can gain insights on how to improve service delivery by analyzing shared data [3, 5, 20, 23]. Part of a rise in demand for data sharing is coming from a demand of data for research purposes. Research derives innovations such as developing new business ideas as well as advancing medicine and technology. For instance, sharing of a hospital's medical records to the public could help researchers to discover a new effective cure for a long-lasting disease or possibly find a way to prevent such disease in its early stages. [3, 4, 27].

There is more good that could come out of meeting the demand for data sharing, but there are concerns relating to privacy. Sharing data concerning individuals threatens to disclose sensitive information [10]. For instance, medical records on a database of a hospital may contain sensitive information that an individual may not want everyone else to know, such as their decision concerning cosmetic surgery or receiving psychiatric services [7, 10]. Organizations are not willing to share their databases for four reasons. Firstly, these organizations value data they possess. Secondly, sharing original data may put them in a position which does not allow them for future data collection even if disclosed information is not considered sensitive

[15]. Thirdly, these organizations could be sued for violating agreements of protecting personal information that they have put forward during data collection. Lastly, these organizations are required to conform with regulations and legislations that govern data sharing such as the Protection of Personal Information Act (POPI Act) of South Africa [7] or the General Data Protection Regulation (GDPR) of the European Union [30], the European Data Protection Law and Restrictions on International Data Flows [29] and the Health Information Portability and Accountability Act (HIPPA) which is a national standard for the privacy of patient medical information [9]. So it is in every organization's interest to protect data in their possession concerning individuals [4, 15, 7].

This creates a dilemma situation between allowing sharing of original data which raise concerns relating to privacy and not allowing it which prevent realization of benefits which could be reaped from sharing data. So organizations would only publicly share their databases if they have confidence that privacy of individuals would not be compromised.

There are several techniques which can be applied to transform databases to protect individuals identities. For instance, a random perturbation technique called Differential Privacy ensures that a removal or addition of a record from a database does not change the outcome of analysis, thereby hiding statistical nature of added or removed record [12]. For instance, a salary bracket of an individual could be revealed if adding such an individual to a database lead to a higher salary average. But perturbation techniques produce misreported databases which cannot be used in applications where data should contain zero false observations such as in medical applications [15, 13, 11]. Also random perturbation techniques are involved with adding noise to numerical variables which makes them ineffective for categorical variables [21, 13, 11]. A solution to these problems would be to consider De-identification techniques that involves generalization and suppression.

De-identification does not produce misreported results as random perturbation methods does and is applicable to both numerical and categorical variables. But a naive De-identification of databases is vulnerable to attacks that combines de-identified databases with other publicly available information (known as linking attack) to re-identify records, thereby disclosing sensitive information. A solution to this problem is introducing anonymity to the records of a database using method proposed in [28] called k-anonymization [28, 4].

TABLE 1.1: Original data set.

Names	**Gender**	**Age**	**Zip Code**	**LSPW**	**D S**
Edvin	Male	40	23051	100 SEK	Cancer
Robert	Male	32	70825	200 SEK	Flu
Alva	Female	40	23051	150 SEK	Cancer
Eleanor	Female	38	23038	170 SEK	Typhus
Ebbe	Female	29	81307	195 SEK	Flu
Signe	Female	34	23038	200 SEK	Typhus
Axel	Male	39	23051	150 SEK	Cancer

TABLE 1.2: Anonymized data set.

Gender	**Age**	**Zip Code**	**LSPW**	**D S**
****	[39,40]	23051	100 SEK	Cancer
****	[29,32]	*****	200 SEK	Flu
****	[39,40]	23051	150 SEK	Cancer
Female	[34,38]	23038	170 SEK	Typhus
****	[29,32]	*****	195 SEK	Flu
Female	[34,38]	23038	200 SEK	Typhus
****	[39,40]	23051	150 SEK	Cancer

The k-anonymization method ensures that for every record in the anonymized data, there are at least $k-1$ other records similar to it with respect to a set of attributes called *quasi-identifier* attributes so that it will be impossible for an attacker to identify which record(s) belongs to which person(s). For instance, Table 1.2 is the anonymized version of Table 1.1 satisfying 2-anonymity with respect to Gender, Age and Zip Code attributes. For example, it would be difficult to disclose that Nash spent R100 per week on lottery even if it is publicly known that he is 40 years old and that he lives in the area with the Zip Code 23051. This is because there are three records in Table 1.2 representing Nash, (the first, third and the last record)

and no one can identify which of the three records belongs to Nash (if the original database is not shared).

The strength of privacy increases with the parameter k in the k-anonymization process. This is because no one in the k-anonymized database can be re-identified with probability greater than $\frac{1}{k}$ by linking the anonymized data with other publicly available information alone (linking attack alone) [4].

Quasi-identifier attributes are all variables of a record after removing personally identifying variables together with sensitive variable(s) such as Diagnosis Status (D S) in Table 1.1. These are the attributes that have a potential of revealing sensitive information by combining them with other variables of publicly available information (linking attack) and are referred to as public variables in [16] which includes age, gender, height, weight of individuals. For instance, Gender, Age and Zip Code in Table 1.1 and Table 1.2 form quasi-identifiers of these two tables. The at least k records that are similar form a group called an *equivalence class* [4, 8]. For instance, the first, third and the last record in Table 1.2 form an equivalence class.

FIGURE 1.1: Education taxonomy tree

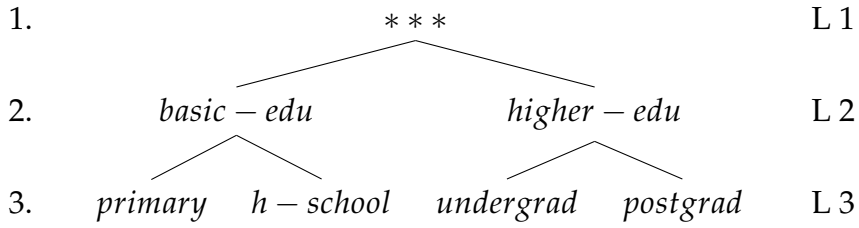

TABLE 1.3: Original transaction data set

Names	**Purchased items**
Nash	Asthma products, Eye products
Pandelani	Asthma products, Eye products, Beef
Katlego	Eye products, Beef
Dineo	Asthma products, Skin products, Beef
Thompho	Asthma products, Beef
Risuna	Eye products, Bread
Thabang	Skin products, Beef

TABLE 1.4: Anonymized transaction data set

Purchased items
(Asthma products, Eye products)
(Asthma products, Eye products) (Beef, Bread, Skin products)
(Asthma products, Eye products) (Beef, Bread, Skin products)
(Asthma products, Eye products) (Beef, Bread, Skin products)
(Asthma products, Eye products) (Beef, Bread, Skin products)
(Asthma products, Eye products) (Beef, Bread, Skin products)
(Beef, Bread, Skin products)

The process of *k*-anonymization involves two operations, data suppression and generalization. Suppression is a process of removing personally identifying variables of records such as names, surnames, social security numbers and email addresses. For instance, Table 1.2 has dropped names from Table 1.1. Other variables which are not personally identifying could also be suppressed if they threaten a privacy requirement given. For instance, Gender of some individuals is replaced with **** from Table 1.1 to Table 1.2. Generalization is a process of changing specific values to general values. For instance, a specific country in Africa could be generalized to just Africa or a specific age number could be generalized to an interval. Generalization could follow a hierarchical system such as the one in Figure 1.1. For instance, *h-school* could be generalized to *basic-edu* if the entry *h-school* threatens to disclose certain individuals in anonymized database. Hierarchical system such as the one in Figure 1.1 is referred to as *taxonomy tree* [8]. Generalization could be *global recoding* or *local recoding*. Every identical value of an attribute would be generalized to the same value in an anonymized database under *global recoding*. *Global recoding* generally produces anonymized databases with higher information loss. An identical value would be generalized to different values under *local recoding*. *Local recoding* produces anonymized databases with less information loss compared to *Local recoding*, but it is computationally expensive to achieve.

k-anonymization should produce minimally distorted databases to ensure that shared databases remain useful. For instance, anonymized medical records should be able to produce the forecasts that there would be an ebola outbreak in the next

ten months if an unanonymized version would produce the same forecasts. So there should be a measurement of the amount of distortion resulting from the anonymization. Distortion is measured using an informational loss metric in most research [4, 8, 15, 24, 21, 17].

Producing minimally distorted databases requires a careful consideration of records before k-anonymization is applied. One way to achieve this is to cluster similar records together first and anonymize clusters individually. The clusters would then be merged together to complete a database. There are two types of clustering algorithms, Divisive/ Partitional and Agglomerative algorithms [18]. All records begin in one cluster and are recursively split based on the dissimilarities (top-down) under Partitional clustering algorithms. Each record starts as a cluster and is recursively merged with other clusters according to the similarities (bottom-up) under Agglomerative clustering algorithms [18].

1.2 Problem statement

It is computationally expensive to process k-anonymization with clustering. It is computationally expensive in a sense that processing both the clustering and the k-anonymization using current algorithms would take long, especially when large datasets are involved. For instance, the execution time of the k-anonymization algorithm which incorporates clustering done in [8] is estimated as $T = \frac{m(m-1)}{2}$ (the proof is done in Section 2.2), where m is a number of records in a data set. Time efficiency of k-anonymization algorithms with clustering have become essential to operations of many academics, governments and industrial institutions. Therefore, this research posed a need for developing faster implementation of k-anonymization algorithms with clustering to improve execution time. This research managed to achieve estimated time of $T = \frac{m(m-Nc)}{2Nc}$ for the new algorithm where, Nc is the number of clusters (the proof is presented in Section 3.5).

1.3 Research Aims and Objectives

1.3.1 Research Aims

This study has aimed to make a significant difference in computational speed. So that, everyday activities which would make use of transformed data would not be delayed if time effective computations is achieved. This research was set to find a way to optimize time computation of the *k*-anonymization algorithm which incorporates clustering done in [8] by introducing parallel implementation. Because, it is ideal to have more time effective algorithms, so that less time is spent waiting for anonymized results especially, when dealing with large data sets.

1.3.2 Objectives

The literature review in Section 2 shows that *k*-anonymization combined with clustering is the ideal way to anonymize data, because of quality results it produces. The objective of this research was to come up with new *k*-anonymization algorithm which incorporates clustering that spends less time processing anonymization. To achieve the aim, the objective of this research was set to identify sections of current algorithm which can be processed independently and concurrently. The identified sections were "parallelly" implemented.

1.4 Research questions

This research sought to address the primary question of what can be done to achieve time effective computation of the *k*-anonymization algorithm which incorporates clustering. The study will sought to answer the following secondary questions

- How could independent sequential implementations of the algorithm done in [8] be changed to run concurrently in order to optimize computation time?
- How will the new algorithm perform in terms of data quality?
- Will the new implementation of the algorithm outperforms previous implementation?

1.5 Limitations

Parallel implementation of algorithms require a computer with certain hardware such as Central Processing Unit (CPU) with multiple cores. The higher the number of cores of CPU the effective will be the parallel implementation. Computers with multiple cores are very expensive and finding a computer with high number of cores had limited this research. The resource used to compute the results was a cluster with a **batch** partition for general purpose use. Up to 60 nodes each with an i7 7700 CPU, and 16GB of RAM and the **ha** partition for high priority runs when you anticipate load shedding may affect your work. Up to 10 nodes each with an i7 7700 CPU, and 16GB of RAM. The nodes of this cluster were configured on a first come, first served basis with a time limit on running the jobs submitted. However, a student could only take a maximum of 10 nodes at a time. The number of nodes that were used on average was five. There is also an issue of CPU overhead which measures the amount of work a computer's CPU can process. This means that even a computer with multiple cores has a limit on how far it can go in terms of the amount of work it can do. For instance, running the new algorithm to process 30 000 records when privacy parameter equals 50 would require 600 processes to be run concurrently which may or may not be possible depending a computer hardware. It is like when you run a parallel computer program, and a computer prompts that it requires an additional CPU core to run that program. [26].

Chapter 2

Literature Review

There are several publications of papers on *k*-anonymization method. The *k*-anonymization algorithms which incorporate clustering has been done in [8, 15, 24, 21, 17, 22, 19, 2]. This section reviews some of these algorithms.

2.1 The *k*-anonymization with clustering review

The *k*-anonymization algorithm incorporating clustering, which allow for a wide range of privacy requirements that a data owner may have, is done in [15] and applied on a transaction database, such as the one shown in Table 1.3. A transaction database is a database which contains a collection of items associated to an individual. The flexibility to allow different privacy requirements helps to avoid unnecessary information loss. Protecting only the items that needs to be protected from disclosure would preserve more information.
For instance, suppose that an attacker only knows that Dineo has bought Asthma products and Skin products which would form privacy requirement $p = \{$Asthma products, Skin products$\}$ in case of Table 1.3. Suppose that k parameter equals 4, this means that at least 4 transaction records of the anonymized Table 1.4 should be associated with both Asthma products and Skin products. The algorithm was used to anonymize the transaction data set in [15] to quality results because it allows users to control generalizations over the items which needed to be generalized. Therefore, avoiding over-generalization [15].

Definition 1 Attribute Diversity:

An attribute diversity of $\mathcal{V}_{a_1}$, represented by $ad\left(\mathcal{V}_{a_1}\right)$ is defined as

$$ad\left(\mathcal{V}_{a_1}\right) = \begin{cases} \frac{\max\left(\mathcal{V}_{a_1}\right) - \min\left(\mathcal{V}_{a_1}\right)}{\max\left(\mathcal{D}_{a_1}\right) - \min\left(\mathcal{D}_{a_1}\right)} & \text{interval values} \\ \frac{|distinct\left(\mathcal{V}_{a_1}\right)|}{|\mathcal{D}_{a_1}|} & \text{discrete values} \end{cases}$$

where a_1 is an attribute with a domain $\mathcal{D}_{a_1}$ and $\mathcal{V}_{a_1} \subseteq \mathcal{D}_{a_1}$ is a subset of values of an attribute a_1 and $\max\left(\mathcal{V}_{a_1}\right)$, $\min\left(\mathcal{V}_{a_1}\right)$, $\max\left(\mathcal{D}_{a_1}\right)$, $\min\left(\mathcal{D}_{a_1}\right)$ represents maximum and minimum value of $\mathcal{V}_{a_1}$ and $\mathcal{D}_{a_1}$ respectively. $|distinct\left(\mathcal{V}_{a_1}\right)|$ is the count of unique values in $\mathcal{V}_{a_1}$ and $|\mathcal{D}_{a_1}|$ is the size of the domain $\mathcal{D}_{a_1}$ [24].

Definition 2 Records Diversity:

Given a set of records $\mathcal{R} \subseteq D$ (D being the original database) over a set of attributes $A = \{a_1, a_2, \ldots, a_m\}$, the records diversity of $\mathcal{R}$ over A, represented by $rd\left(\mathcal{R}, A\right)$ is defined as

$$rd\left(\mathcal{R}, A\right) = \sum_{i=1}^{m} ad\left(\pi_{a_i}\left(\mathcal{R}\right)\right)$$

where $\pi_{a_i}\left(\mathcal{R}\right)$ represents a projection of $\mathcal{R}$ on attribute a_i [24, 25].

Definition 3 Usefulness:

Assume that records of a database D are clustered into a set of clusters $C = \{c_1, c_2, \ldots, c_h\}$ such that $|c_i| \geq k, i \in [1, h]$ and the records of c_i will have equal values of quasi-identifier set after anonymization. The *usefulness* of the database D under the clustering C is defined as

$$usefulness = avg\left(rd\left(c_1, QID\right), \ldots, rd\left(c_h, QID\right)\right)$$

where $rd\left(c_i, QID\right)$ represents records diversity of clusters $c_i, i \in [1, h]$ with respect to the set of quasi-identifier (QID) [24].

Definition 4 Protection:

Assume that records of a database D are clustered into a set of clusters $C = \{c_1, c_2, \ldots, c_h\}$ such that $|c_i| \geq k, i \in [1, h]$ and the records of c_i will have equal values of quasi-identifier set after anonymization. The *protection* of the database D under the clustering C is defined as

$$protection = avg\left(\frac{1}{rd\,(c_1, SA)}, \ldots, \frac{1}{rd\,(c_h, SA)}\right)$$

where $rd\,(c_i, SA)$ represents records diversity of clusters $c_i, i \in [1, h]$ with respect to the set of sensitive attributes (SA) [24, 25].

The *usefulness* of a database D is measured by the average of the records diversity of all clusters over a set of quasi-identifier (QID). The lower the value of *usefulness* the closer are the records of a cluster $c_i, i \in [1, h]$ on the values of quasi-identifier attributes. Therefore, there would be little modification needed to achieve k-anonymity. Hence, an anonymization of a database D would preserve more information. The *protection* of a database D is measured by the average of the inverse of records diversity of all clusters over a set of sensitive attributes (SA). The lower the value of *protection* the further apart are the values of records on sensitive attributes. Therefore, it would imply that sensitive attributes are highly protected in the sense of diversity [24].

The results of anonymizing Table 1.1 to form Table 1.2 had only managed to protect disclosure of the first sensitive attribute Lottery Spendings Per Week (LSPW), it did not protect Diagnosis Status (DS) of any of the persons whose age is either 39 or 40 (Nash, Katlego and Thabang) if an attacker knows their age. This is because each of the members in the equivalence class {****, [39,40], 23051} corresponding to Gender, Age and Zip Code is diagnosed with cancer which makes it obvious to deduce their status. This is because the attribute DS lacks diversity. To guard against this kind of threat of disclosing sensitive information, [24] has introduced a metric which measure diversity in the set of sensitive attributes and is provided in **Definition 4**. Suppression of the Zip Codes of Pandelani and Thompho has made their individual records useless for applications which would require zip codes to be meaningful. So [24] also suggested a metric to control against sharing databases that contain suppressed attributes which are required to be meaningful in analysis, and it is provided in **Definition 3**. An algorithm in [24] was built around these

two metrics to generate clusters, grouping records simultaneously with respect to similarities of quasi-identifier attributes and dissimilarity of records with respect to sensitive attribute values [24].

Definition 5 Weighted Hierarchical Distance:

Let h be the height of a hierarchical system such as the height of education domain in Figure 1.1 and let $1, 2, \ldots, h-1, h$ be domain levels from the most general to the most specific level, respectively. For instance, level one (L1) is the most general level domain and level 3 (L3) is the most specific level domain of education in Figure 1.1. Let the weight between two domain levels i and $i-1, i \in [2, h]$ be represented by $\omega_{i,i-1}$. The weighted hierarchical distance of a generalization is defined as

$$WHD\,(p,q) = \frac{\sum_{i=q+1}^{p} \omega_{i,i-1}}{\sum_{i=2}^{p} \omega_{i,i-1}}$$

when an attribute value is generalized from level p to q, where $p > q$ [21]. The two different weights $(\omega_{i,i-1})$, the *uniform weight* and the *height weight* are well discussed in [21]. The *uniform weight* is shown to have a shortcoming, failing to capture that generalizations near or at root node has a higher distortion of information than generalizations near or at leaves nodes. This is because it assigns equal weights to all generalization levels, irrespective of whether it is near a root node or near a leaf node. The *height weight* is capable of capturing a difference in distortion across different levels of generalization [21].
The *height weight* is given by

$$\omega_{i,i-1} = \frac{1}{(i-1)^{\beta}}$$

where $i \in [2, h]$ and $\beta \geq 1$ is a real number that would be given by a user. It is clear to see that the *height weight* would be able to capture the difference in distortion across different levels. This is because the lower the level (near or at the root node), the higher the value of the *height weight* would be. Conversely the higher the level (near or at the leaf node), the lower would be the value of the *height weight* [21].

Definition 6 Distortions of Generalizations of Records:

Let $r = \{a_1, a_2, \ldots, a_m\}$ be a record with attributes a_1 to a_m and $r' = \{a'_1, a'_2, \ldots, a'_m\}$ be a generalized record of r. Let a level of an attribute domain of $a_i, i \in [1, m]$ in a hierarchical system such as the one in Figure 1.1, be represented by $level\left(a_i\right), i \in [1, m]$. The distortion of a generalization is defined as

$$distort\left(r, r'\right) = \sum_{i=1}^{m} WHD\left(level\left(a_i\right), level\left(a'_i\right)\right).$$

For instance, suppose that hierarchical system of Gender, Age and Zip Code in Table 1.1 follows $\{Male\ or\ Female, ****\}, \{DD, [DD, DD], **\}$ and $\{DDDDD, DDDD*, DDD**, DD***, D****, *****\}$ repectively. Given that $\beta = 1$, distortion of the first record from Table 1.1 to Table 1.2 on Gender, Age and Zip Code attributes would be $WHD = 1.00$, $WHD = 0.67$ and $WHD = 0$ respectively. Therefore, distortion of the generalization of the first record from Table 1.1 to Table 1.2 is 1.67. Distortion of an anonymized database $\tilde{D}$ of D would be a summation over a distortion of all records between the two databases [21].

Definition 7 Closest Common Generalization:

Let $r_1 = \{a_1, a_2, \ldots, a_m\}$ and $r_2 = \{a_1, a_2, \ldots, a_m\}$ be two different records of a database D. Suppose r_{12} is the closest common generalization of r_1 and r_2. Then r_{12} is defined as

$$r_{12}^i = \begin{cases} a_1^i & \text{if } a_1^i = a_2^i \\ \text{the value of the closest common ancestor} & \text{otherwise} \end{cases}$$

where a_1^i, a_2^i and r_{12}^i are the values of the i-th attribute in the records r_1, r_2 and r_{12} respectively. For instance, the value of the closest common ancestor in the education attribute of two different records, one taking a value of *undergrad* and the other taking a value of *postgrad* would be *higher-edu* if the two records follow a hierarchical system in Figure 1.1 [21].

Definition 8 Distance Between Two Records:

Let $r_1 = \{a_1, a_2, \ldots, a_m\}$ and $r_2 = \{a_1, a_2, \ldots, a_m\}$ be two different records of a database D and r_{12} be a closest common generalization of r_1 and r_2. Then distance between the two records is defined as

$$dist\,(r_1, r_2) = distort\,(r_1, r_{12}) + distort\,(r_2, r_{12})\,.$$

For instance, suppose that hierarchical system of Gender, Age and Zip Code in Table 1.1 follows $\{Male\ or\ Female, ****\}, \{DD, [DD, DD], **\}$ and $\{DDDDD, DDDD*, DDD**, DD***, D*****, ******\}$ respectively. Now, the closest common generalization of the first and the second would be a record $r = \{****, 40, 23051\}$ on Gender, Age and Zip Code respectively. Therefore, the distance between these two records would be 1 given that $\beta = 1$ on the *height weight*. This is because distortion only happened on the Gender attribute, since Age and Zip Code values would be kept intact [21].

Definition 9 Distance Between Two Clusters:

Let c_1 be a cluster containing n_1 identical records r_1 and c_2 be another cluster containing n_2 identical records r_2 and r_{12} be a closest common generalization of r_1 and r_2. Then the distance between the two clusters is defined as

$$dist\,(c_1, c_2) = n_1 \times distort\,(r_1, r_{12}) + n_2 \times distort\,(r_2, r_{12})\,.$$

This is a metric which controls whether two clusters should be merged. Two clusters would only be merged if a record r_{12} to which the two clusters are going to be generalized to gives a minimal distance from their original clusters [21].

Definition 10 Stub and Trunk of a Cluster:

Suppose a small cluster c_1 needs to be generalized with a large cluster c_2 to achieve k-anonymity. A cluster c_2 needs to be divided into two divisions, a *stub* and a *trunk*, if $|c_1| < k$ and $|c_1| + |c_2| \geq 2k$. A *stub* would contain a number of records needed by c_1 to satisfy k-anonymity, which is $k - |c_1|$ records. A *trunk* would contain the remaining number of records $|c_2| - k + |c_1|$, which also satisfy k-anonymity with no need of generalization. It should be noted that there would be no *stub* or a *trunk*

if $|c_1| + |c_2| < 2k$ in which case all the records from two clusters would just be generalized together to achieve k-anonymity. There are two cases in which distance between two clusters would be calculated to see if merging the two clusters would result in minimal information loss [21].

$$\begin{cases} \text{If } |c_1| + |c_2| < 2k & \textit{use } \textbf{\textit{Definition 9}} \\ \text{If } |c_1| + |c_2| \geq 2k & \textit{distance between } c_1 \textit{ and stub of } c_2. \end{cases}$$

Now [21] developed an algorithm to process k-anonymization to ensure that quality anonymized databases are produced. The distance metric in **Definition 9** was used to control merging of clusters [21]. The distance would be calculated based on a hierarchical system such as the one shown in Figure 1.1 and it would ensure that a cluster is merged to a cluster which introduces minimal distortion. Incorporating a clustering technique helps to keep some records intact, and avoids unnecessary information loss [21].

Definition 11 Distance Between Two Numeric Values:

Let $\mathcal{D}$ be a domain of a numerical attribute of a database. The distance between two numerical values $v_1, v_2 \in \mathcal{D}$ is defined as

$$dis_N(v_1, v_2) = \frac{|v_1 - v_2|}{|\mathcal{D}|}$$

where $|\mathcal{D}|$ is the size of the domain $\mathcal{D}$, calculated by a difference between maximum and minimum value of $\mathcal{D}$.
A database can also contain attributes which are not numerical and which cannot be ordered to any specific order [8]. A distance between two categorical attribute values which has hierarchical relation can be calculated as in **Definition 8**, except that it can also be evaluated without a weight function defined.

Definition 12 Distance Between Two Categorical Values:

Let $\mathcal{D}$ be a domain of categorical attribute of a database and $\mathcal{T}_{\mathcal{D}}$ be a hierarchical system (*taxonomy tree*) like the one in Figure 1.1. Then a distance between two categorical values $v_1, v_2 \in \mathcal{D}$ is defined as

$$dis_C(v_1, v_2) = \frac{H(\wedge(v_1, v_2))}{H(\mathcal{T}_{\mathcal{D}})}$$

where $\wedge(v_1, v_2)$ is the *subtree* of a full *taxonomy tree* rooted at a common ancestry of v_1 and v_2 and H measures the height of a given tree. For instance, the distance between *undergrad* and *postgrad* in Figure 1.1 would be $\frac{1}{2}$ and 1 for distance between *primary* and *undergrad* [8].

Definition 13 Distance Between Two Records In Absence Of Weight Function:

Let $QID = \{n_1, \ldots, n_m, c_1, \ldots, c_n\}$ be quasi-identifiers set. The n_i, $i \in [1, m]$ representing numerical attributes and c_j, $j \in [1, n]$ representing categorical attributes. Then a distance between two records r_1 amd r_2 denoted by $\bigtriangledown(r_1, r_2)$ is defined as

$$\bigtriangledown(r_1, r_2) = \sum_{i=1}^{m} dis_N(r_1[n_i], r_2[n_i]) + \sum_{j=1}^{n} dis_C(r_1[c_j], r_2[c_j])$$

where $r_1[Atr]$ represents a value of r_1 in attribute Atr [8].

Definition 14 Information Loss:

Let $c = \{r_1, r_2, \ldots, r_k\}$ be a cluster of records with quasi-identifiers set $QID = \{n_1, \ldots, n_m, c_1, \ldots, c_n\}$. The n_i, $i \in [1, m]$ representing numerical attributes and c_j, $j \in [1, n]$ representing categorical attributes. Let $\mathcal{T}_{C_j}$ represent a *taxonomy tree* for categorical attributes c_j, $j \in [1, n]$. Let Max_{n_i} and Min_{n_i} represent maximum and minimum value in cluster c with respect to attribute n_i, $i \in [1, m]$ and let $\cup_{c_j}$ be a union of values in c with respect to attribute c_j, $j \in [1, n]$. Then information loss that would result from generalizing c, represented by $IL(c)$ is defined as

$$IL(c) = |c| \times \left(\sum_{i=1}^{m} \frac{Max_{n_i} - Min_{n_i}}{|n_i|} + \sum_{j=1}^{n} \frac{H\left(\wedge\left(\cup_{c_j}\right)\right)}{H\left(\mathcal{T}_{C_j}\right)} \right)$$

where $|c|$ is the number of records in c, $|n_i|$ is the domain size of an attribute, $\wedge\left(\cup_{c_j}\right)$ is a subtree at a lowest common ancestor of every value in $\cup_{c_j}$ and $H\left(\mathcal{T}_{\mathcal{C}_j}\right)$ is the height of the taxonomy tree with respect to attribute c_j, $j \in [1, n]$ [8].

Definition 15 Total Information Loss:

Let $C = \{s_1, s_2, \ldots, s_h\}$ be a set of equivalence classes resulting from anonymizing database D to a database $\tilde{D}$. Then total information loss of $\tilde{D}$ is defined as

$$Total\_IL\left(\tilde{D}\right) = \sum_{i=1}^{h} IL\left(s_i\right).$$

The k-anonymization algorithm done in [22] is an improvement of the algorithm done in [8] regarding how to handle outlier records. The algorithm in [8] would randomly select a record from a database D to form a cluster and select a furthest record from the randomly selected record to form a second cluster, and so forth. The furthest record would most likely be an outlier record if a database D contains outliers. The clusters formed on outliers would incur maximal information loss. The algorithm in [22] would randomly select $K = \lfloor \frac{|D|}{k} \rfloor$ from sorted D by quasi-identifiers to form clusters c_i, $i \in [1, K]$. Then a first record from the remaining records of D would be removed and allocated to a cluster that gives minimal distance each time as long as D still contains records. Building a cluster on an outlier record to contain k records would be impossible since the distance metric would not allow it. Clusters would need to be adjusted after all records are removed from D. There would be clusters with more than k records and others with less than k records. The records out of clusters which have more than k and which are further from the centroid of the cluster than the rest of the records would be removed from that cluster and be donated to closer clusters which have less records. However, outlier clusters would need to be deleted in the end if it is not worth to merge them with the donated records [22]. This algorithm would produce quality anonymized databases, but suppression of the outlier cluster should also be considered as a loss of information.

2.1.1 Algorithm targeted for improvement

Algorithm 1 Greedy k-member clustering
Input: Database D and a parameter k
Output: An anonymized database $\tilde{D}$.

1: **if** $(|D| \leq k)$ **then**
2: **return** D
3: **else**
4: $\tilde{D} = \{\}$
5: R = randomly selected record from D
6: **while** $(|D| > k)$ **do**
7: r = furthest record from R
8: $c = r$
9: $D = D - \{r\}$
10: **while** $(|c| \leq k)$ **do**
11: $r = find\_best\_record\,(D, c)$
12: $c = c \cup \{r\}$
13: $D = D - \{r\}$
14: **end while**
15: $\tilde{D} = \tilde{D} \cup c$
16: **end while**
17: **while** $(|D| \neq 0)$ **do**
18: r = randomly selected record from D
19: $c = find\_best\_cluster\,(\tilde{D}, r)$
20: $c = c \cup \{r\}$
21: $D = D - \{r\}$
22: **end while**
23: $\tilde{D} = \cup_{c \in \tilde{D}} H\,(c)$
24: **return** $\tilde{D}$
25: **end if**

Now [8] did the clustering Algorithm 1 for anonymization of data in order to minimize information loss defined in **Definition 14** & **15** of an anonymized database $\tilde{D}$ resulting from D. Algorithm 1 check if the number of records in a database D is less or equal to a parameter k from which it would return a database D intact in line 2. Otherwise, the algorithm would randomly select a record R from D in line 5 and then find a furthest record from R based on distance **Definitions 11, 12** & **13** in line 7. A record in line 7 would become the only member of a cluster c in line 8 and it would then be removed from database D in line 9. Thereafter, records that

introduce minimal information loss to a cluster c would be found one by one based on the information loss **Definitions 14** & **15** in line 11. A record r in line 11 would be inserted to a cluster c in line 12 each time it is found and removed from database D in line 13. The algorithm would then insert a cluster c to a database $\tilde{D}$ once it contain k number of records in line 15. The process from line 7 to line 15 would be repeated until a database D is remaining with records less or equal to k at which point a record r would be randomly selected in line 18. Then a close cluster to a record r in line 18 would be found in line 19 according to **Definitions 14** & **15** to be inserted to that cluster in line 20 and the record r in line 18 would be removed from a database D each time in line 21. The process from line 18 to 21 would be repeated until a database D is empty. The clusters in $\tilde{D}$ would then be generalized by a function H in 23 if clustering alone is not sufficient to satisfy k-anonymity per cluster. Then the algorithm would return anonymized database $\tilde{D}$ in line 24. It is easy to see in Algorithm 1 that using clustering before generalization would produce quality anonymized databases [8].

2.2 Estimation of execution time of the old algorithm

The observation is that algorithm 1 spends most of its time selecting a record from the data set D of size m, one at a time until it reaches $|D| = k$. The execution time T, as the data set size $|D|$ decreases by one record every iteration is estimated as

$$T = (m-1) + (m-2) + \ldots\ldots + k.$$

Considering the worst case scenario (when $k = 1$), the execution time is given by

$$T = (m-1) + (m-2) + (m-3) + \ldots\ldots + 3 + 2 + 1 \tag{2.1}$$

and rearranging is also true, that is

$$T = 1 + 2 + 3 + \ldots\ldots + (m-3) + (m-2) + (m-1) \tag{2.2}$$

adding (2.1) to (2.2) gives

$$2T = m + m + m + m + \ldots\ldots + m \qquad (m-1)\ \textit{times}$$

and the execution time T solves to

$$T = \frac{m\,(m-1)}{2} \tag{2.3}$$

which is an estimation of the execution time of the algorithm 1.

Chapter 3

Research Methodology

This section outlines the methodology used in producing the results. A description of research design is given, including data collection, and data splitting which forms part of preprocessing. The parallel implemented algorithm is described as well as the description of metrics to measure quality of transformed data. The computational time of the proposed algorithm is estimated.

3.1 Research design

This research outlines a way to optimize computational time of the algorithm done in [8]. The main focus is to find sequential sections of the algorithm which has processes that can be computed independently and make them run concurrently in parallel. The algorithm under discussion is algorithm 1 in this research report. Algorithm 1 finds a record which becomes a cluster and append that cluster until it contains k records. It repeats this process until no further clusters can be generated. This research proposed to find $K = \lfloor \frac{n}{k} \rfloor$ ($n = |D|$, the data size of D data set and k being a privacy parameter) records of a data set which have maximal distances between each other and those records would form bases for all clusters which could be generated. The bases would be appended simultaneously until each cluster contains k records. For instance, suppose that a database D contains 200 records and that a parameter k equals 7, then the algorithm would find 28 records from D that have maximal distance between each other. Then, the 28 records would form 28 clusters to be appended simultaneously until each cluster contains k records. Thereafter, any remaining records would be appended to clusters that introduce minimal information loss when merged together. The results of the algorithm in [8] will be compared to the results of the newly proposed algorithm in Section 3.4.

3.2 Data and data preprocessing

The data sets used for running the experiments of this research was the Adult data set from UC Irvine Machine Learning Repository [6] and the Restaurant Scores-Lives Standard data set from https://healthdata.gov/search/type/dataset. The Adult data set is considered the most suitable data set for evaluating k-anonymization algorithms performance. The algorithms were used on only the nine attributes used in [8] in the Adult data set. The attributes used in [8] were *age, work class, education, marital status, occupation, race, gender and native country* including the sensitive attribute *salary*. The first eight attributes chosen are the *quasi-identifiers*, they were chosen because they can be used to link individuals to their sensitive attributes (*salary* in this case) when combined together. Only five attributes were selected for the same reason from the Restaurant Scores data set, *business name, business postal code, inspection score, inspection type* including the sensitive attribute *risk category*. Both data sets had to be cleaned, removing records with missing values and converting some attributes to an easy to work with data type.

Finding the *centroids* of clusters using the new algorithm 2 in Section 3.4 by the traditional way of using distance metric had proven to take longer. This is because the algorithm would have to go through every record of the remaining records of a data set each time, to find a furthest record from the already generated *centroids*. This was a drawback in time efficiency. This problem was resolved by splitting the data sets into different groups containing similar records before the algorithm was applied, so that the algorithm would only calculate the distance between the already generated centroids and one record per those groups of records to find a new *centroid* (see Figure 3.1 for more details). A viable option was to combine the use of k-means and k-modes clustering [1] to perform data splitting since the data set had both numerical and categorical variables. The k-means clustering was used to split the data along numerical variables and the k-modes was used along categorical variables [1]. The k-means clustering randomly select data points which are used as the beginning centres for all the clusters to be generated and then perform a repetitive process of optimizing distances between the centres of the clusters until the centres are stabilized [14]. Each data point out of the remaining ones will be allocated to a closer centre, and a new centre for each cluster will be the average of

all the points in a cluster taken along attributes. This process will be repeated until there is no change in allocation of data points to centres [14]. The *k*-modes clustering uses the same idea except that the centres are determined by modes along each attribute of a data point, but the process is the same.

3.3 Methods

The results in this report were obtained using a cluster with the specifications described in Section 1.5. The algorithms were implemented and run in Python 3 platform, version 3.7.3. Five nodes were chosen each time a code of new algorithm was run.

3.4 Proposed algorithm

Algorithm 2 Greedy k-member clustering with parallelism
Input: Splitted data set D and a parameter k
Output: An anonymized database $\tilde{D}$.

if $(|D| \leq k)$ **then**
  return D
else
  $\tilde{D} = \{\}$, $\tilde{C} = \{\}$
  $m = |D|$, $N_c = \lfloor \frac{m}{k} \rfloor$ ▷ N_c is the number of clusters to be generated
  $r_1, r_2, \ldots, r_{N_c} = find\_N_c\_records\,(D, N_c)$ ▷ with maximal distances from each other
  $C =$ a list of clusters $c_1, c_2, \ldots, c_{N_c}$ formed on $r_1, r_2, \ldots, r_{N_c}$
  $D = D \setminus \{r_1, r_2, \ldots, r_{N_c}\}$
  while $(|D| \geq k)$ **do**
    while $\left(\sum_{i=1}^{|C|} |c_i|_{c_i \in C} < |C| \times k\right)$ **do**
      $C, records = find\_and\_append\_best\_records\,(D, C)$ ▷ simultaneously
      $D = D \setminus records$
      for c *in* C **do**
        if $|c| = k$ **then**
          $\tilde{C}\,.\,append\,(c)\,,\ C\,.\,remove\,(c)$
        end if
      end for
    end while
  end while
  while $(|D| \neq 0)$ **do**
    $n = |D|$
    allocate the n records in D to $\{r_1, r_2, \ldots, r_n\}$
    $C' = find\_best\_clusters\,\left(\tilde{C}, \{r_1, r_2, \ldots, r_n\}\right)$ ▷ simultaneously
    append $c_i \in \tilde{C},\ \forall i \in C'$ with the corresponding $r \in \{r_1, r_2, \ldots, r_n\}$
    $D = D \setminus \{r_1, r_2, \ldots, r_n\}$
  end while
  $\tilde{D} = \cup_{i \in \tilde{C}} Anonymize\,(i)$
  return $\tilde{D}$
end if

Function $find\_N_c\_records\,(D, N_c)$
Input : D, a set of groups of similar records and a parameter N_c (# of clusters to be found)
Output : a set of records with maximal distances between each other

1. records = [ ] an empty list
2. records . append (randomly selected record r in D), $D = D \setminus \{r\}$
3. **while**($|\text{records}| < N_c$) **do**
   $q = |records|$
   $rec = r$ from any of the groups in D $s.t.$ $\sum_{i=1}^{q} \bigtriangledown (r, r_i)$ is maximal
   where $r_i \in$ records and $\bigtriangledown$ is given by definition 13 in Section 2.1
   records . append(rec), $D = D \setminus \{rec\}$
   end while
   NB : $\sum_{i=1}^{q} \bigtriangledown (r, r_i)$ is computed as a pool in parallel
4. return records

Function $find\_and\_append\_best\_records\,(D, C)$
Input : a set of records D and a list of clusters C
Output : updated set of clusters and a set of records appended to clusters

1. $best =$ empty array of size N_c
2. $\forall\, j, i$ in $zip\,(C, 1 : N_c)$
3. $best[i] = (j, r \in D\ s.t.\ IL\,(c_i \cup r)$ is minimal) *simultaneously computed*
4. $rec =$ unique pairs (cluster, record) taken from $best$ along the 2^{nd} coordinate.
5. append the records in rec to their corresponding clusters in C (only once).
   $records =$ a list of all records in rec taken along the 2^{nd} coordinate.
6. return $C, records$

Function $find\_best\_clusters\,(C, \{r_1, r_2, \ldots, r_n\})$
Input : a set of clusters C and a set of records $\{r_1, r_2, \ldots, r_n\}$
Output : a set of clusters indices C' of size n such that $\forall\ i\ in\ C'$
$IL\,(c_i \cup r)$ is minimal, $r \in \{r_1, r_2, \ldots, r_n\}$.

1. best = empty array of size n
2. $\forall$ i in $1 : n$
3. best$[i] = c.index()$, $c \in C\ s.t.\ IL\,(c \cup r_i)$ is minimal, $r_i \in \{r_1, r_2, \ldots, r_n\}$
4. return best

FIGURE 3.1: Functions in new algorithm explained

The following box explains some of the lines in the new algorithm.

- **Line 12**: $D = D \setminus records$, (this line is executed efficiently as follows)
  $index = \textbf{DataFrame}(records).index()$ $index()$ is a built-in function for pandas **DataFrame** to collect indexes.
  $D = np.delete(D, index, axis = 0)$
  NB: All the records are deleted together at once. The same applies to **Line 8**.

- The reason **Lines 20 to 26** were included in the algorithm
  These lines are there to ensure that the resulting anonymized table has a minimal information loss so far as possible. For instance, appending all the remaining records to one cluster could cause that particular cluster to be fully suppressed when anonymization is applied especially, if the attributes of the added records are too dissimilar to the existing records.

- **Line 27** the function *Anonymize*
  This function is used to anonymize a given cluster by applying generalization and/or suppression.

FIGURE 3.2: Lines in new algorithm explained

From the last two functions in the previous page, two algorithms can be developed based on how the information loss is calculated. If the algorithm is run with the classification metric described in Section 3.6.3, the algorithm becomes *greedy k member with cm* and *greedy k member* otherwise.

3.5 Estimation of execution time of the new algorithm

The observation is that algorithm 2 spends most of its time selecting Nc (the number of clusters to be generated) records from dataset D at a time for $\lfloor \frac{m}{k} \rfloor$ iterations (m being the number of records in D and k being a privacy parameter) until D contains less than Nc records. The execution time of algorithm 2 is estimated as

$$T = (m - Nc) + (m - 2Nc) + (m - 3Nc) + (m - 4Nc) \tag{3.1}$$

the sequence going for $\lfloor \frac{m}{Nc} \rfloor$ terms in above series in (3.1). Applying the arithmetic series formula

$$\sum_{k=1}^{n} (a + (k-1)\, d) = \frac{n}{2} (2a + (n-1)\, d)$$

to (3.1) with $a = m - Nc$, $d = -Nc$ and $n = \frac{m}{Nc}$ would give

$$T = \frac{m\,(m - Nc)}{2Nc}. \tag{3.2}$$

3.6 Analysis

3.6.1 Distance metric

Distance between two records was computed using the distance metric used in [8], given by

$$\bigtriangledown\,(r_1, r_2) = \sum_{i=1}^{m} dis_N\,(r_1[n_i], r_2[n_i]) + \sum_{j=1}^{n} dis_C\,(r_1[c_j], r_2[c_j])$$

where $dist_N$ is defined in **Definition 11**, $dist_C$ is defined in **Definition 12**, and $r_i[Atr]$ represents a value of a record r_i with respect to the attribute Atr. This metric is used in line 6 to find records with maximal distances between each other (*the centroids*) and to find a furthest record to form a new cluster in the serial Algorithm 1. The new algorithm 2 would randomly selects a record from a data set D to form a basis for a first cluster and finds a record furthest from the first cluster to form a second cluster. Thereafter it finds a furthest record from the first two clusters formed. A record to form a basis for a third cluster in the proposed algorithm is computed simultaneously. That is, the algorithm would go through the records of data set D

one by one. Thereby, calculating the distances between a record picked from D and the two clusters already formed simultaneously. Thereafter, taking a sum of those distances and a record that gives highest sum of distances will be a basis for a third cluster and so on. This process will be repeated until N_c records having maximal distances between each other are found.

3.6.2 Information loss metric

Information loss which would result from generalizing a formed cluster is given in **Definition 14** [8] and defined by

$$IL\left(c\right) = |c| \times \left(\sum_{i=1}^{m} \frac{Max_{n_i} - Min_{n_i}}{|n_i|} + \sum_{j=1}^{n} \frac{H\left(\wedge\left(\cup_{c_j}\right)\right)}{H\left(\mathcal{T}_{C_j}\right)} \right)$$

where $IL\left(c\right)$ is defined and explained in **Definition 14**.

3.6.3 Classification metric

The k anonymization method is mostly focused on the *quasi-identifier* attributes, forgetting that correlation between *quasi-identifier* variables and the target variables (*sensitive* attributes) needs to be preserved as well. Considering the correlation between *quasi-identifier* variables and the target variables can ensure that classification models trained on the transformed data performs better. So, it is important that the anonymization process does not lose the discrimination of *sensitive* variables using *quasi-identifier* variables. All this can be controlled using the Classification metric (CM) defined by

$$CM = \sum_{i=1}^{m} \frac{penalty\left(row\ i\right)}{m}$$

where m is the number of records and $penalty\left(row\ i\right) = 1$ if $row\ i$ is completely suppressed or if the target value of row i is different from the majority target value in the equivalence class [8].

The information loss metric in line 3 of the last two functions in Figure 3.1 are evaluated in the following way when CM is applied

if majority class of target value in c == class of row r:
$IL = IL(c \cup r)$
else:
$IL = IL(c \cup r) \; + \; class \;\; penalty$
NB : The *class penalty* above is not entirely **CM** (defined in the previous page), **CM** is calculated for all the records in an equivalence class whereas the *class penalty* is the penalty given if the *sensitive* value of a record to be added to a cluster is not equal to majority *sensitive* value in the cluster or if the new record would cause the cluster's records to be suppressed when k-anonymization is applied.

3.6.4 Discernibility metric

The quality of transformed data can also be measured based on the size of each equivalence class generated [4]. This quality measure is done based on the indistinguishability of records with each other in a equivalence class. Every unsuppressed record in the equivalence class of size i will get a penalty of i and every suppressed record will get a penalty of the size of the data set D, $|D|$. This is to represent that a suppressed record cannot be distinguished from any other record in the data set, hence the penalty of $|D|$. The discernibility metric is defined by

$$DM = \sum_{\forall\ c\ s.t.\ |c| \geq k} |c|^2 + \sum_{\forall\ c\ s.t.\ |c| < k} |c||D|$$

where c is an equivalence class produced by anonymization, k is privacy parameter and D represents data set [4].

Chapter 4

Results and Discussion

This section presents the core findings of this research derived from the methods and techniques provided in the methodology section (in Section 2). The results and corresponding discussions are presented in separate sub sections.

4.1 Time complexity of the algorithms

TABLE 4.1: Time complexity of the old and new algorithms respectively

Old algorithms estimated time	**New algorithms estimated time**
$T = \frac{m(m-1)}{2}$	$T = \frac{m(m-Nc)}{2Nc}$

Table 4.1 shows that time complexity of the old and of new algorithms is the same, that is, $O\left(m^2\right)$. It can be observed nonetheless, that there is a significant improvement in computational time from the old to the new algorithm. The time that the new algorithm spends processing k-anonymization depends on the size of data as well as, number of clusters (Nc) to be generated. It is clear that, the higher the number of clusters to be generated, the lower the numerator expression of the estimated time for the new algorithm. It can also be seen that the denominator will be high when the number of cluster to be generated (Nc) increases. So a decrease in the numerator coupled with an increase in the denominator of the time complexity of the new algorithm will result in shorter time for the new algorithm. Let us do more mathematical analysis of the time complexity of the new algorithm by taking a limit as $Nc \rightarrow m$, a data set size.

$$\lim_{Nc \to m} T = \lim_{Nc \to m} \frac{m\,(m - Nc)}{2Nc} = 0 \tag{4.1}$$

therefore, as $Nc \to m$, the time complexity will be in $O\,(0)$. This means that time for the new algorithm follows a linear relation as $Nc \to m$. It should be noted that this can only be achieved if we have an unlimited number of CPU cores and before CPU overhead kicks in. Nonetheless, these are good results.

4.2 Data quality results

4.2.1 Information loss results

This section reports on the results of information loss against different k values for the old and new algorithm. Both results of algorithms have two parts, one implemented with classification penalty to reduce classification error and the other without the penalty. As Figure 4.1 illustrates, the results of the new algorithm are close to those of the old algorithm even though all the centroids of the new algorithm are estimated before the anonymization process is carried out.

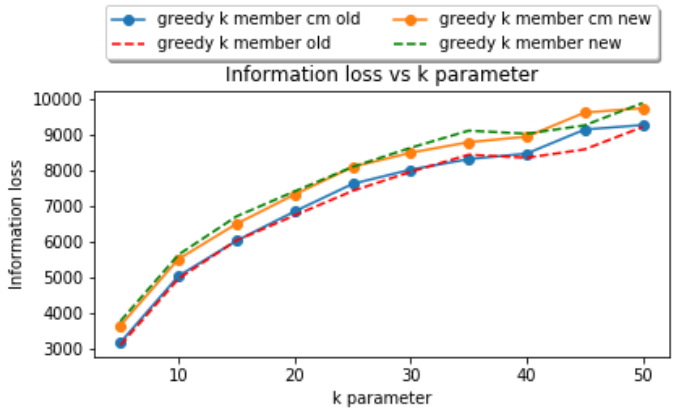

FIGURE 4.1: New vs old algorithms, 2000 records used. k-anonymization on the Adult data set.

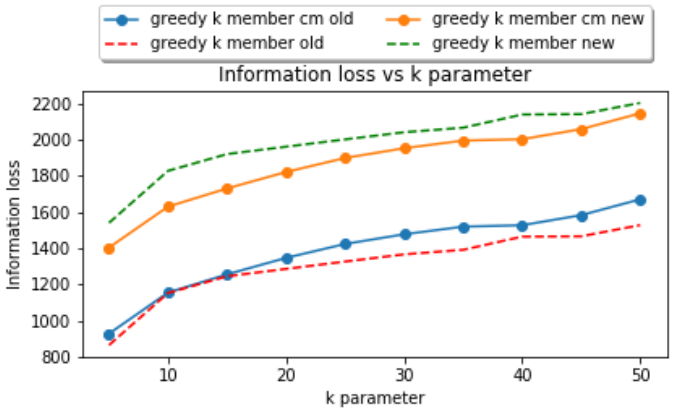

FIGURE 4.2: New vs old algorithms, 2000 records used. k-anonymization on the Restaurant Scores data set.

The two graphs in Figures 4.1 and 4.2 shows an expected trend of the resulting information loss. It is expected that information loss will decrease when the k parameter is increased. This is because more records will be required to form an equivalence class. As more records are required per an equivalence class, the chance of finding the records to be similar will decrease. Figure 4.1 shows that the new algorithms are considerably performing the same as the old algorithms when the Adult data set is used. However, Figure 4.2 indicates that the new algorithms produce less quality anonymized data sets when Restaurant Scores data set is used. This could have been caused by the presence of outlier records in the Restaurant Scores data set. The new algorithms could have also contributed to higher information loss. This is because they do not go through every record of a data set to determine a cluster. Thus, it could have found similar records to be bases for clusters because it finds bases for clusters on the group level of the pre-processed groups of records. There is a significant gap between the new algorithms and old algorithms, however, the results of the new algorithms could be improved when Restaurant Scores data set is increased.

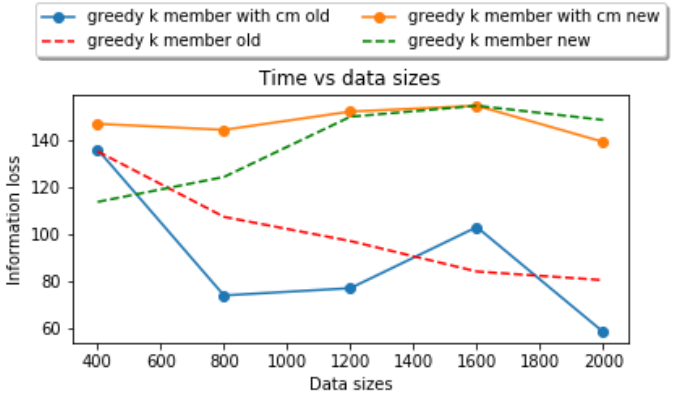

FIGURE 4.3: New vs old algorithms, $k = 15$. k-anonymization on the Adult data set.

Figure 4.3 shows that the new algorithms have under performed across all different data sizes. The resulting information loss from the new algorithms is looking high compared to old algorithms. This is because the new algorithms do not go through all the records of a data set to find a set of records with maximal distances in between. The new algorithms search through clusters of records already formed during the preprocessing stages of data, evaluating a distance of just one record in a cluster (*centroid* of a cluster) from the already formed clusters to form a basis for a new cluster. That is how records with maximal records are found and it could happen that some of the clusters formed are not that far apart from each other. Thus maximal distance is determined on the group level across all groups, not on the record level. This should not just rule out the new algorithms because the maximum number of records used here was just 2000 records. The results could change when more records are used, thereby increasing a chance of picking records which indeed have maximal distances between them. So the new algorithms could still be a way to go when faced with large data sets. The old algorithm with *cm* also shows an unexpected trend, whereby the information loss drastically increase as data size increases. However, this only occurred with the old algorithm with classification error penalty. The trend with the algorithm without classification error penalty gradually decrease as data sizes increase as expected. Figure 4.3 shows the increase of information loss between data sizes 1200 and 1800. This could have happened because the increased data sets also included the outlier records. This

means that different results can be seen with the increase of the data size as it it seen when the data size reached 2000 records. Similarly both of the new algorithms displayed an increase information loss when data size increased from 800 records to 1600 records. Again, this could be because the additional records are not similar enough to reduce information loss.

4.2.2 Classification results

This section reports on the results implemented according to the classification error defined in Section 3.6.3 against different k values for the old and new algorithm.

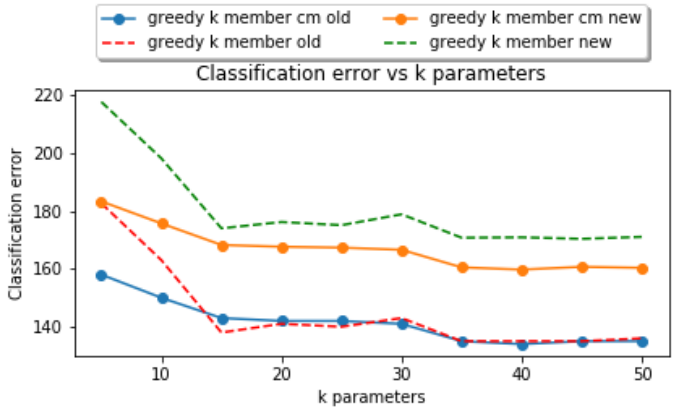

FIGURE 4.4: New vs old algorithms, 2000 records used. k-anonymization on the Adult data set.

Figure 4.4 illustrates the results of classification error in the anonymized data sets. This is done to determine if the correlation between the *quasi-identifier* attributes and *sensitive* attributes is preserved after the anonymization. That is to ensure that any model fitted on an anonymized data set does not misinterprets the association between features and target values because the anonymization had misrepresented them. It is expected that *greedy k member* with *cm* algorithms will show a smaller classification errors than that of without *cm* algorithms. This is because the algorithms with *cm* ensures that the records that will preserve a correlation between *quasi-identifier* attributes and *sensitive* attributes are grouped in the same cluster to ensure less classification error of the anonymization process. Figure 4.4 confirms the expectations, *greedy k member* with *cm* is mostly under *greedy k member* without

cm in the old algorithms and the same is seen for the new algorithms. The graphs representing new algorithms look as if there were just translated from those representing the old ones. The new algorithms could have a factor of records per an equivalence class penalized for having a class not belonging to the majority class. This gap between the old algorithms and new algorithm could the solved by running the new algorithms when the Adult data set is increased.

4.2.3 Discernibility results

This section reports on the results implemented according to the discernibility penalty defined in Section 3.6.4 against different k values for the old and new algorithm. Figure 4.5 shows that the new algorithms are producing equivalence classes with just slightly a higher generalization than the old algorithms.

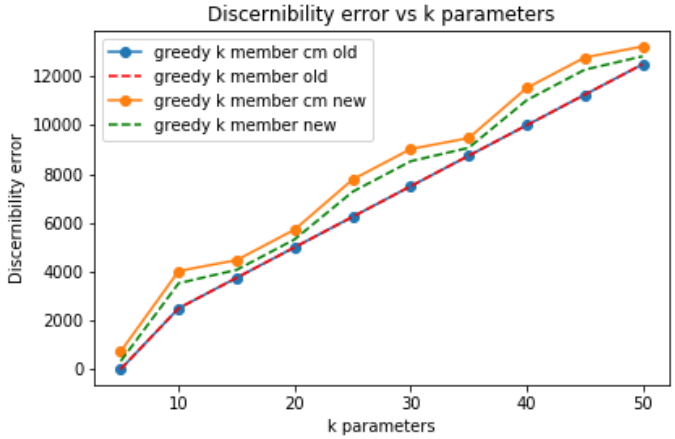

FIGURE 4.5: New vs old algorithms, 2000 records used. k-anonymization on the Adult data set.

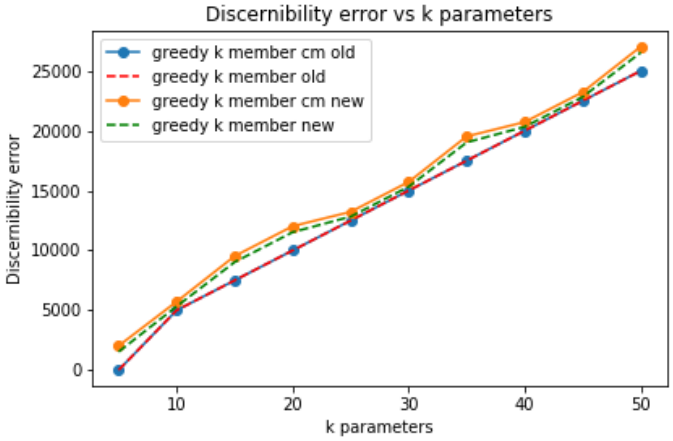

FIGURE 4.6: New vs old algorithms, 2000 records used. k-anonymization on the Restaurant Scores data set.

It is expected that the discernibility penalty will increase with an increase in k parameter. This is because a higher k value means more records will be contained in an equivalence class. As the k parameter increases the chance of finding those many similar records to avoid the discernibility penalty will decrease. Hence, more records will be suppressed during the anonymization which causes a higher discernibility penalty. All the algorithms are doing almost equally the same. There is no big gap between the old algorithms and the new algorithms. The discerninility substantiate in favour of choosing to use new algorithms provided they are faster, since there is no much difference between the old and new algorithms. The old algorithms formed a sharp straight line just after k equals 10 until k equals 50. This indicates consistency in the old algorithms. There has been slight ups and downs in the discernibility trend of the new algorithms. They are not perfectly consistence. However, the new algorithms performed considerably good compared to the old ones. This phenomenon is displayed in the both data sets (the Adult and Restaurant Scores data sets used to produce Figure 4.5 and 4.6).

4.2.4 Execution efficiency results

This section reports on the results of execution efficiency for the old and new algorithm. Figure 4.7 illustrates the results of execution efficiency against different privacy parameter k values and Figure 4.9 shows the results of execution efficiency against increasing data sizes.

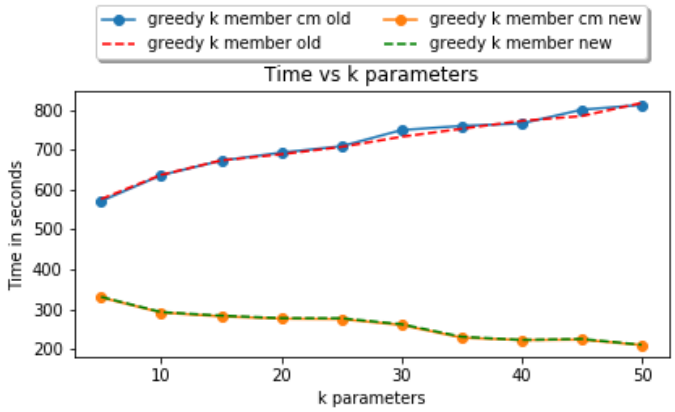

FIGURE 4.7: New vs old algorithms, 2000 records used. k-anonymization on the Adult data set.

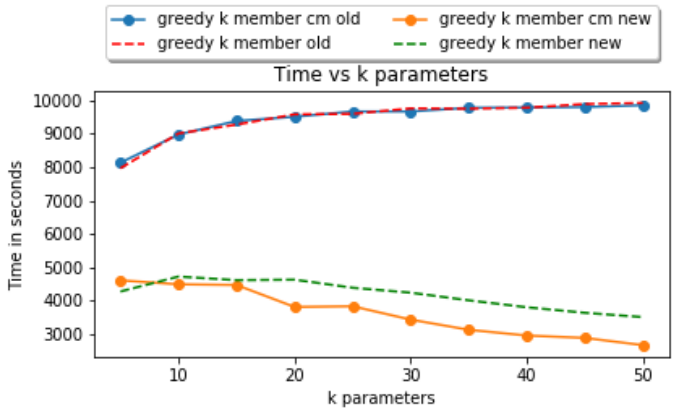

FIGURE 4.8: New vs old algorithms, 2000 records used. k-anonymization on the Restaurant data set.

Figure 4.7 shows the results of execution time of both the old and new algorithms. The results are indicating that the old algorithms time is increasing slightly as k values increase. The opposite would have been expected, since increasing k values imply less number of clusters are going to be generated as more records would be contained per cluster. The execution time of the old algorithms should have been decreasing because the old algorithms would have not been run a higher number of times trying to find furthest records to form bases for clusters. However, this should not be much of a time cost since the execution time increases slightly. For

the new algorithms, expectations are vice versa based on Section 4.1, only increased number of clusters to be processed should speed up the process. Thus increasing k values keep a data size constant should lead to a slower process. This is not quite the case in Figure 4.7. This could be because the CPUs reached maximum capacity when $k = 50$ and it becomes beyond the capacity when $k < 50$. That could be the why execution time decrease towards $k = 50$. This is an indication that better performance of parallel algorithms is on the expense of more hardware. More spending is needed to achieve these better results. Despite all these, new algorithms are worth considering if the resources needed for parallel implementations can be afforded. The similar results are shown when the algorithms are applied to the Restaurant Scores data set. There is a surprising gap in the new algorithms between *greedy k member* with *cm* and *greedy k member* without . The *greedy k member* with *cm* is below the *greedy k member* without *cm* from about k equals 15 to k equals 50. This is not expected at all. It could be easily explained if it was the other way around. Thus, the *greedy k member* with *cm* would have been spending longer time trying to find records with less classification error. However, this is not a big gap, the results are still acceptable.

TABLE 4.2: The k parameters, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms with cm and speedup of the algorithms using the Adult data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
5	2000	400	571.181705	329.113756	1.735515
10	2000	200	636.358030	291.385797	2.183902
15	2000	133	672.798351	281.731618	2.388083
20	2000	100	693.001153	276.130467	2.509687
25	2000	80	708.888634	275.010510	2.577678
30	2000	66	749.562634	260.007537	2.882850
35	2000	57	759.993992	228.311315	3.328762
40	2000	50	766.133721	222.213255	3.447741
45	2000	44	800.671835	223.388243	3.584217
50	2000	40	812.734867	209.599446	3.877562

TABLE 4.3: The k parameters, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms without cm and speedup of the algorithms using the Adult data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
5	2000	400	576.273045	330.678345	1.742700
10	2000	200	637.002202	292.500321	2.177783
15	2000	133	673.332521	283.543891	2.374703
20	2000	100	688.696373	277.199234	2.484482
25	2000	80	707.219636	277.003489	2.553107
30	2000	66	732.685907	261.996731	2.796546
35	2000	57	752.772049	230.557809	3.265003
40	2000	50	772.561507	222.563217	3.471200
45	2000	44	784.700247	225.000896	3.487543
50	2000	40	817.692189	210.400257	3.886365

TABLE 4.4: The k parameters, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms with cm and speedup of the algorithms using the Restaurant data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
5	2000	400	8136.705367	4610.843100	1.764689
10	2000	200	8979.413684	4492.804740	1.998621
15	2000	133	9385.222571	4472.443350	2.098455
20	2000	100	9509.750844	3814.56077	2.493013
25	2000	80	9669.319397	3836.138090	2.520587
30	2000	66	9668.179233	3435.548820	2.814159
35	2000	57	9780.571090	3130.927080	3.123858
40	2000	50	9786.024261	2958.794650	3.307436
45	2000	44	9801.639511	2887.651210	3.394329
50	2000	40	9855.907436	2672.054780	3.688512

TABLE 4.5: The k parameters, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms without cm and speedup of the algorithms using the Restaurant data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
5	2000	400	7969.051421	4277.715080	1.862922
10	2000	200	9010.103295	4724.961240	1.906916
15	2000	133	9275.896900	4616.808680	2.009158
20	2000	100	9581.877943	4631.601920	2.068804
25	2000	80	9598.860974	4389.635910	2.186710
30	2000	66	9760.922584	4242.380020	2.300813
35	2000	57	9752.753864	4013.043650	2.430264
40	2000	50	9781.843643	3805.587140	2.570390
45	2000	44	9891.562376	3639.966360	2.717487
50	2000	40	9925.304324	3510.391560	2.827407

Now it is time to analyze what happens when the algorithms are run with different number of clusters to be generated, represented by a column **In parallel** in Table 4.2 and 4.3. These results are performed when data size is kept fixed at 2000 records, just changing the k values. The speedup column in both tables represent the time spent by an old algorithm in seconds divided by the time spent by a new algorithm. The results would indicate that the new algorithm performed better if the resulting speedup is greater than 1.0. A speedup less than 1.0 would indicate the otherwise. Both the algorithms started by running 400 processes in parallel while the old algorithms run single processes in sequential order. The speedup there was close to 2 on both tables (one for the algorithms with and the other without classification penalty) which implies that new algorithms performed about two times better than the old algorithms. As k increases the speedup started to increase. An increase in speedup as k increases could be influenced by CPUs computational capacity. Performing 400 tasks in parallel with a limited number of CPU cores could put a strain on CPUs performing ability. Hence parallel implementation is only able to double the performance. But the new algorithms begin to speed up as the number of tasks to be performed in parallel decreases. The speedup increased to close to 4

with 40 processes in parallel in both tables. This implies that new algorithms performed about four times better than old algorithms. The number of records used were 2000, and this size would be considered not enough to make conclusions. However, the algorithm used is scalable and deterministic. Therefore, these results could be considered good. Similar results are displayed when the algorithms are run on Restaurant Scores data set. Those results are presented in Table 4.4 and 4.5.

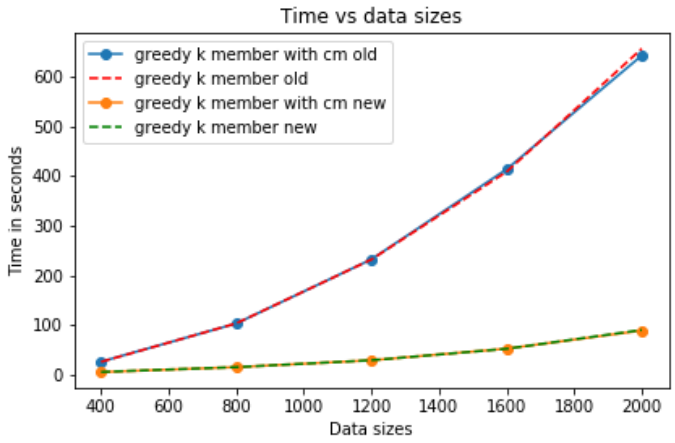

FIGURE 4.9: New vs old algorithms, $k = 15$. k-anonymization on the Adult data set.

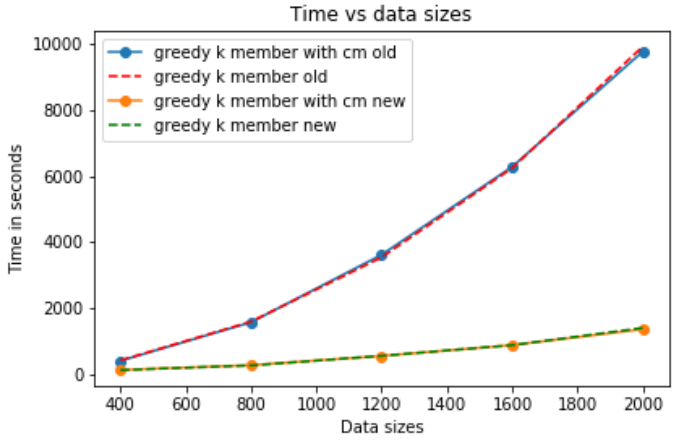

FIGURE 4.10: New vs old algorithms, $k = 15$. k-anonymization on the Restaurant Scores data set.

Figure 4.9 shows the results of performing k-anonymization using different data sizes when the privacy parameter k is kept constant at $k = 15$. The new algorithms

clearly performed better than the old algorithms. The old algorithms computational time increased significantly as data size increases whereas, the new algorithms execution time increased linearly as the data size increases. This confirms the result of the analysis given in Section 4.1, that the execution time of the new algorithm follows a linear relation. This is because of keeping the privacy parameter constant and increasing data sizes which imply that the number of clusters to be generated (N_c) would increase when data size increase. Hence, the linear relation being displayed coinciding with the phenomenon shown in Section 4.1. The new algorithms, the one with classification metric embedded in calculating information loss and the one without classification metric, only spent a longest time of 100 seconds as data size increased. On the other hand, the old algorithms processing time increased to over 600 seconds as data size increased. This shows that the new algorithms are a good alternative considering execution time efficiency. Similar results are displayed when the algorithms are run on Restaurant Scores data set. Those results are presented in Figure 4.10.

Now it is time to analyze what happens when the algorithms are run with different number of clusters to be generated, represented by a column **In parallel** in Table 4.6 and 4.7. This has been done when keeping k constant, increasing data size thereby, increasing the number of clusters to be generated (number of processes to be performed in parallel). Table 4.6 and Table 4.7 shows the results obtained for the algorithms with and without classification metric respectively. The speedup columns of both tables were obtained through dividing the time spent (in seconds) of the old algorithms with the time spent (also in seconds) by the new algorithms. This implies that the results of a speedup greater than 1.0 indicates that a new algorithm has done better than the old algorithm with time and a speedup less than 1.0 indicates that a new algorithm has been outperformed by an old algorithm in computational time. A speedup much greater than one indicates that a new algorithm has done much much better compared to an old algorithm whereas, a speedup less than one and much closer to zero indicates that an old algorithm has done better compared to new algorithm. The speedup started with about 5 for both the algorithms with *cm* and without *cm* when the number of clusters to be generated was 26. This indicates that the new algorithms performed five times better than the old algorithms. The results became much better for the new algorithms as the number of clusters to be generated increased through 53 to 80 ,and the speedup increased to about 8. The

new algorithms were performing eight times better than the old algorithms right there.

TABLE 4.6: The k parameter, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms with cm and time ratios of the algorithms using the Adult data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
15	400	26	26.381023	5.410331	4.876046
15	800	53	103.127995	15.286600	6.746300
15	1200	80	232.296105	29.093985	7.984334
15	1600	106	413.582887	52.469085	7.882411
15	2000	133	642.453671	88.919717	7.225098

TABLE 4.7: The k parameter, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms without cm and time ratios of the algorithms using the Adult data set respectively.

k	**Data size**	**In parallel**	**Old**	**New**	**Speedup**
15	400	26	25.921747	5.660250	4.579612
15	800	53	103.395834	12.261554	8.432523
15	1200	80	231.997544	29.164813	7.954707
15	1600	106	408.435670	52.240755	7.818334
15	2000	133	656.272815	90.049173	7.287938

But the speedup started to decrease slightly when the number of clusters to be generated started to increase to 106 through to 133. This could be explained by being limited with the number of CPU cores and by the CPUs reaching maximum capacity (CPU overhead). The number of records considered here were only 2000 records, but the algorithm is scalable and deterministic. So overall perspective, the results show that the new algorithms are the better options when the efficiency is to be considered.

TABLE 4.8: The k parameter, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms with cm and time ratios of the algorithms using Restaurant Scores data set respectively.

k	**Data size**	**In parallel**	**Old cm**	**New cm**	**Speedup**
15	400	26	393.142198	117.091113	3.357575
15	800	53	1563.985791	262.597906	5.955820
15	1200	80	3612.160354	541.899908	6.665733
15	1600	106	6282.018567	871.950459	7.204559
15	2000	133	9776.940818	1352.463780	7.228985

TABLE 4.9: The k parameter, data sizes, number of processes in parallel for new algorithm, time (in seconds) of old and new algorithms without cm and time ratios of the algorithms using Restaurant Scores data set respectively.

k	**Data size**	**In parallel**	**Old**	**New**	**Speedup**
15	400	26	406.946638	113.498638	3.585476
15	800	53	1583.327386	253.101216	6.255708
15	1200	80	3535.046209	545.111196	6.485000
15	1600	106	6245.738673	869.069553	7.186696
15	2000	133	9929.466668	1392.329680	7.131549

Similar results are displayed when the algorithms are run on Restaurant Scores data set. Those results are presented in Table 4.8 and 4.9.

4.3 Summary

Parallel implementation of *k*-anonymization incorporating clustering algorithms was designed in Section 3 to cut on computational time of running anonymization, at the same time maintaining quality anonymized data sets. This section presented the results of running the experiments, testing old algorithms against new algorithms. Section 4.1 illustrated the mathematical results of time efficiency of the new and the old algorithms. The results had shown that time complexity of the new algorithms is the same as that of the old algorithms. Nonetheless, the new algorithms have improved the execution efficiency (the results are shown in Table 4.1). There is also an aggregation step in line 12 in the new algorithm where the records appended to the clusters are deleted. However, this step can be considered insignificant because it can be executed efficiently (with little time, see Figure 3.2). It has been shown that the new algorithms follow a linear relation as the number of clusters (Nc) to be generated approach a data set size. Sections 4.2.1, 4.2.2 and 4.2.3 presented the results of data quality of the anonymized data sets of new algorithms against old algorithms. The results between the old and new algorithms were similar in some cases, but the old algorithms performed better that the new algorithms. However, the new algorithms could still produce good results when data sets are very large, meaning a higher chance of find similar records. Section 4.2.4 presented the results of computational time. The number of records used were 2000 which could be considered not enough to draw conclusions however, the algorithms are scalable and deterministic. Therefore, it can be concluded that the new algorithms vastly out performed the old algorithms especially, when CPU overhead was not an issue.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Given that data publicizing raises concerns relating to privacy of disclosing personal sensitive information of data subjects through association techniques, literature has indicated that *k*-anonymization method can be used to ensure that data subjects cannot be re-identified. However, *k*-anonymization incurs information loss, which is why clustering of similar records is performed before anonymization is applied. Processing both the clustering and *k*-anonymization using current algorithms is computationally expensive. This research provided a solution of introducing parallel implementation to the algorithms done in [8] to solve the problem. The new algorithms were out performed by the old algorithms when it comes to data quality of anonymized data, however, the results of the new algorithms were still acceptable. The new algorithms performed considerably faster when it comes to execution efficiency (the number of records used were 2000, but the algorithms are scalable and deterministic which means resource can be scaled up when more records are added, the results will be similar), but there was always an issue of CPU overhead which limited the new algorithms. The new algorithms were found to be reliable when it comes to execution efficiency. It is safe to say that the new algorithms performed better with computational time, except when the performance slowed down due to CPUs overhead.

5.2 Future Work

This research was implemented on a cluster computer. This research can be extended to

- be implemented on a distributed system to see if it will further improve the results.
- explore a better way to handle outlier records.
- run *k*-anonymization over the internet in cloud.
- run the implementation on much larger data sets.

Bibliography

[1] *Abhishek Kar. "All about analytics explains–Using K-modes for clustering categorical data"*. https://analyticsdefined.com/using-k-modes-clustering-categorical-data/. Accessed: 2019-07-30.

[2] Gagan Aggarwal et al. "Achieving anonymity via clustering". In: *ACM Transactions on Algorithms (TALG)* 6.3 (2010), p. 49.

[3] Trevor J Barnes. "Big data, little history". In: *Dialogues in Human Geography* 3.3 (2013), pp. 297–302.

[4] Roberto J Bayardo and Rakesh Agrawal. "Data privacy through optimal k-anonymization". In: *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*. IEEE. 2005, pp. 217–228.

[5] David Beer. "How should we do the history of Big Data?" In: *Big Data & Society* 3.1 (2016), p. 2053951716646135.

[6] Catherine L Blake and Christopher J Merz. *UCI repository of machine learning databases, 1998*. 1998.

[7] JG Botha, Mariki M Eloff, and I Swart. "The effects of the PoPI Act on small and medium enterprises in South Africa". In: *2015 Information Security for South Africa (ISSA)*. IEEE. 2015, pp. 1–8.

[8] Ji-Won Byun et al. "Efficient k-anonymization using clustering techniques". In: *International Conference on Database Systems for Advanced Applications*. Springer. 2007, pp. 188–200.

[9] Steven L Clause et al. "Conforming to HIPAA regulations and compilation of research data". In: *American Journal of Health-System Pharmacy* 61.10 (2004), pp. 1025–1031.

[10] Terence Craig and Mary E Ludloff. *Privacy and big data: the players, regulators, and stakeholders*. " O'Reilly Media, Inc.", 2011.

[11] Fida Kamal Dankar and Khaled El Emam. "The application of differential privacy to health data". In: *Proceedings of the 2012 Joint EDBT/ICDT Workshops*. ACM. 2012, pp. 158–166.

[12] Cynthia Dwork. "Differential privacy". In: *Encyclopedia of Cryptography and Security* (2011), pp. 338–340.

[13] Cynthia Dwork and Adam Smith. "Differential privacy for statistics: What we know and what we want to learn". In: *Journal of Privacy and Confidentiality* 1.2 (2009), pp. 135–154.

[14] Dr. Michael J. Garbade. *Understanding K-means Clustering in Machine Learning, 2018*. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. 2018.

[15] Aris Gkoulalas-Divanis and Grigorios Loukides. "Utility-guided Clustering-based Transaction Data Anonymization." In: *Trans. Data Privacy* 5.1 (2012), pp. 223–251.

[16] Jacob Goldberger and Tamir Tassa. "Efficient anonymizations with enhanced utility". In: *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*. IEEE. 2009, pp. 106–113.

[17] Somesh Jha, Luis Kruger, and Patrick McDaniel. "Privacy preserving clustering". In: *European Symposium on Research in Computer Security*. Springer. 2005, pp. 397–417.

[18] Erik L Johnson and Hillol Kargupta. "Collective, hierarchical clustering from distributed, heterogeneous data". In: *Large-Scale Parallel Data Mining*. Springer, 2000, pp. 221–244.

[19] Md Enamul Kabir, Hua Wang, and Elisa Bertino. "Efficient systematic clustering method for k-anonymization". In: *Acta Informatica* 48.1 (2011), pp. 51–66.

[20] Jay Lee, Hung-An Kao, and Shanhu Yang. "Service innovation and smart analytics for industry 4.0 and big data environment". In: *Procedia Cirp* 16 (2014), pp. 3–8.

[21] Jiuyong Li et al. "Achieving k-anonymity by clustering in attribute hierarchical structures". In: *International Conference on Data Warehousing and Knowledge Discovery*. Springer. 2006, pp. 405–416.

[22] Jun-Lin Lin and Meng-Cheng Wei. "An efficient clustering method for k-anonymization". In: *Proceedings of the 2008 international workshop on Privacy and anonymity in information society*. ACM. 2008, pp. 46–50.

[23] Steve Lohr. "The age of big data". In: *New York Times* 11.2012 (2012).

[24] Grigorios Loukides and Jianhua Shao. "Capturing data usefulness and privacy protection in k-anonymisation". In: *Proceedings of the 2007 ACM symposium on Applied computing*. ACM. 2007, pp. 370–374.

[25] Ashwin Machanavajjhala et al. "l-diversity: Privacy beyond k-anonymity". In: *22nd International Conference on Data Engineering (ICDE'06)*. IEEE. 2006, pp. 24–24.

[26] Elizabeth Mott. *What Is CPU Overhead?, 2019*. https://smallbusiness.chron.com/cpu-overhead-37464.html. 2019.

[27] Jeffrey D Sachs and John W McArthur. "Technological advancement and long-term economic growth in Asia". In: *Technology and the new economy, edited bY: Bai, E.-E. and Yuen, C.-W., MIT Press, Cambridge, MA, USA* (2002), pp. 157–185.

[28] Pierangela Samarati and Latanya Sweeney. "Generalizing data to provide anonymity when disclosing information". In: *PODS*. Vol. 98. Citeseer. 1998, p. 188.

[29] Paul M Schwartz. "European data protection law and restrictions on international data flows". In: *Iowa L. Rev.* 80 (1994), p. 471.

[30] Mahsa Shabani and Pascal Borry. "Rules for processing genetic data for research purposes in view of the new EU General Data Protection Regulation". In: *European Journal of Human Genetics* (2017), p. 1.