TFRP: An Efficient Microaggregation Algorithm for Statistical Disclosure Control

Chin-Chen Chang[1,2], Yu-Chiang Li[2], and Wen-Hung Huang[3]

[1]Department of Information Engineering and Computer Science,
Feng Chia University, Taichung 40724, Taiwan, R.O.C.
[2]Department of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi 62102, Taiwan, R.O.C.
E-mail: {ccc, lyc}@cs.ccu.edu.tw
[3]Institute of Information Systems and Applications,
National Tsing Hua University,
101, Section 2, Kuang-Fu Rd., Hsinchu, 30013, Taiwan, R.O.C.
E-mail: csie0916@yahoo.com.tw

Abstract. Recently, the issue of Statistic Disclosure Control (SDC) has attracted much attention. SDC is a very important part of data security dealing with the protection of databases. Microaggregation for SDC techniques is widely used to protect confidentiality in statistical databases released for public use. The basic problem of microaggregation is that similar records are clustered into groups, and each group contains at least k records to prevent disclosure of individual information, where k is a pre-defined security threshold. For a certain k, an optimal multivariable microaggregation has the lowest information loss. The minimum information loss is an NP-hard problem. Existing fixed-size techniques can obtain a low information loss with $O(n^2)$ or $O(n^3/k)$ time complexity. To improve the execution time and lower information loss, this study proposes the Two Fixed Reference Points (TFRP) method, a two-phase algorithm for microaggregation. In the first phase, TFRP employs the pre-computing and median-of-medians techniques to efficiently shorten its running time to $O(n^2/k)$. To decrease information loss in the second phase, TFRP generates variable-size groups by removing the lower homogenous groups. Experimental results reveal that the proposed method is significantly faster than the Diameter and the Centroid methods. Running on several test datasets, TFRP also significantly reduces information loss, particularly in sparse datasets with a large k.

Keywords. Microaggregation, disclosure control, k-anonymity, database security

1. Introduction

Recent developments in data mining techniques have enabled the rapid and efficient discovery of hidden knowledge from very large databases [14]. Users employing such techniques for databases can acquire sensitive knowledge [3]. Since many types of current research heavily depend on statistical data that protect the confidentiality of personal information [1, 11, 22], statistical disclosure control (SDC) has become an important data security issue [7, 16, 18].

Many government agencies and commercial organizations need to collect and analyze data about individuals to support their research activities. The collected individual data (micro-data) contain confidential information of individuals, such as income or type of disease. Each individual j is assigned a data vector V_j, which contains two kinds of variables: key variables, which identify the individual, and sensitive variables, which contain sensitive information about the individual [9].

Therefore, the direct release of databases for study may incur security risks. The challenge for SDC is to modify data in order to provide sufficient protection of sensitive information without seriously damaging the information used for data mining purposes [10].

Database managers often remove explicit identifiers, such as names and phone numbers, to protect privacy. A released database that suppresses identifying attributes is not sufficient to preserve privacy. Linked together, public databases can narrow down or even identify individuals. An example in [19] demonstrates that users can re-identify individuals from anonymous medical data by linking to an external voter list, using non-identifiable attributes such as zip code, birthday, and gender.

One way to protect individuals is to mask the released database that achieves k-anonymity. A release provides k-anonymity protection for $k > 1$ if each entity in the database is indistinguishable from at least $k - 1$ other entities in the database [21]. When an attribute value appears k or more times in a database, it is difficult to re-identify individuals by linking to an external database. Several researchers use generalization and suppression techniques to implement k-anonymity [2, 17, 19, 21]. However, generalization and suppression approaches cause high information loss, especially on numerical attributes [8, 10].

Microaggregation, which satisfies *k*-anonymity [8, 10], is a widely used SDC technique for numerical data and has been extended for categorical data [10, 23]. Instead of releasing actual values of individual records, microaggregation clusters records into groups, and each attribute value of records in one group is replaced by the centroid of the group. For a certain *k*, each group contains at least *k* records to satisfy *k*-anonymity, where the confidentiality of each individual is protected.

The problem of microaggregation differs from the classical clustering problem. In microaggregation, each group has at least *k* data points (data vectors) rather than to assign the number of groups. Given a security parameter *k*, an optimal microaggregation has minimum information loss (or highest data quality) at the fixed security level. Although an efficient algorithm exists for optimal univariate microaggregation [12], multivariate microaggregation has been proven as an NP-hard problem [18]. Therefore, several heuristic methods have been proposed for multivariate microaggregation [9, 15, 20].

An optimal microaggregation has no group containing more than 2*k*-1 records since each group with size ≥ 2*k* can be partitioned in order to further reduce information loss [9]. Although each group is limited to a number of members between *k* and 2*k*-1, the possible partition number still grows exponentially with the number of data vectors [15]. Existing heuristic microaggregation methods can be categorized into two classes: fixed-size microaggregation [9] and variable-size microaggregation [15].

The Diameter and the Centroid methods are two well-known fixed-size microaggregation techniques. Both have low information losses, but the time complexities of Diameter and Centroid are $O(\frac{n^3}{k})$ and $O(n^2)$, respectively, which are too high to practice for a large database. Therefore, this study proposes a two-phase algorithm for microaggregation: the efficient Two Fixed Reference Points (TFRP) method. In the first phase, TFRP employs several techniques such as pre-computing, median-of medians, and partial distance to efficiently reduce the running time to $O(\frac{n^2}{k})$. In the second phase, TFRP scatters the members of the low homogenous groups to the nearest groups to improve data quality.

The rest of this paper is organized as follows: Section 2 introduces the background and the relative Diameter and Centroid methods. Section 3 then describes

the proposed Two Fixed Reference Points (TFRP) algorithm, which is a two-phase method, and analyzes its time complexity. In addition, Section 4 provides experimental results and evaluates the performance of the proposed algorithm. Finally, Section 5 presents conclusions.

2. Related Work

2.1. General remarks

Consider a statistical database with n records and p numerical attributes. Each record is a p-dimensional data point (data vector) in a p-dimensional space. Microaggregation involves combining n data points to form g groups of at least k size, where each data point belongs to exactly one group, called the k-partition [9]. Since each vector is replaced by the group centroid before publication, similar points appear in the same group; this results in low information loss. The squared Euclidean distance is widely used to measure the homogeneity within a group. Equation (1) describes the squared Euclidean distance between two p-dimensional vectors, x and y. The distance value between two vectors indicates the value of their squared Euclidean distance, except where noted.

$$d(x, y) = \|x - y\|^2 = \sum_{i=1}^{p} (x(i) - y(i))^2 , \quad (1)$$

where $x(i)$ and $y(i)$ represent the attribute values of the i-th dimension of x and y, respectively. Equation (2) shows the within-group squared error (GSE) of a group G_i.

$$GSE(G_i) = \sum_{j=1}^{n_i} d(x_{ij} - \bar{x}_i) , \quad (2)$$

where n_i is the element number in the i-th group G_i, $n_i \geq k$, x_{ij} is the j-th element in G_i and $\bar{x}_i$ is the centroid of G_i. The optimal microaggregation is to reach the minimum total sum of the within-group squared errors (SSE). That is,

$$SSE = \sum_{i=1}^{g} GSE(G_i) = \sum_{i=1}^{g} \sum_{j=1}^{n_i} d(x_{ij} - \bar{x}_i) , \quad (3)$$

where g is the number of generated groups and $\sum_{i=1}^{g} n_i = n$. The total sum of square errors (SST) is as follows:

$$SST = \sum_{i=1}^{g} \sum_{j=1}^{n_i} d(x_{ij} - \bar{x}) , \quad (4)$$

where $\bar{x}$ is the centroid of the n vectors. The measure of information loss (IL) is defined as $IL = SSE/SST$. A low IL value signifies higher data quality of a released database.

Statistical agencies typically use the fixed-size technique to implement microaggregation [9]. A microaggregation method usually results in low information loss by fixing each group size to k. The Diameter and Centroid methods are two well-known fixed-size microaggregation methods and two of the best approaches to achieve low information loss [9, 15]. A concise description is as follows:

2.2. Diameter and Centroid methods

In the Diameter method, the pair of initial points (x_r, x_s) with the longest squared Euclidean distance is selected. Then, two groups are formed with size k around x_r and x_s, respectively. One group, G_r, initially contains only one vector x_r, and in each of the k-1 iterations, the vector closest to the centroid of G_r is added to G_r. The other group, G_s, which contains x_s, is similarly formed. The process iteratively selects a pair of maximum distance vectors among the rest of vectors and forms two groups until less than k vectors remain. Each remaining vector is added to its nearest group. The time complexity of the Diameter method reaches $O(\frac{n^3}{k})$ [15].

Instead of clustering two groups around the two extreme vectors in each iteration, the Centroid method selects a furthest vector from the centroid. Centroid, therefore, reduces the time complexity to $O(n^2)$ and still obtains a low information loss [15].

2.3. Minimum spanning tree microaggregation

If the size of each group generated by clustering methods (such as [13, 24]) can be contained the size of each group, such clustering methods may be good heuristic methods for microaggregation. The partitioning minimum spanning tree (MST) is one of the most used methods for clustering. Therefore, Laszlo and Mukherjee proposed a variable-size minimum spanning tree method for microaggregation [15].

They remove edges from the MST in the order of decreasing length. To satisfy the group size constraint, the adapted algorithm cuts the selected edges if the separation produces two admissible sub-trees, which both contain at least k points; otherwise, the edge is skipped. Thus, each resulting tree is a cluster and conforms the requirement of the minimum group size constraint.

The MST partitioning algorithm may generate several large groups, which have size $\geq 2k$. To reduce the information loss, users can employ one of the fixed-size

methods to further separate from the oversize groups. However, the MST partitioning method has higher information loss than that of Diameter, unless the distribution of records in the micro-data set is well-separated natural clusters. Furthermore, the running time of MST is longer than that of Centroid.

3. Two Fixed Reference Points Method

This study proposes the Two Fixed Reference Points (TFRP) method to speed the process performance and reduce information loss. TFRP is a two-phase algorithm. In the first phase, TFRP uses a novel fixed-size algorithm to shorten the running time efficiently. In the second phase, TFRP reduces the number of groups generated by the first phase to improve the data quality. Furthermore, the post-processing technique of the second phase can be applied to any microaggregation method to reduce information loss.

We describe the two-phase algorithm as follows. In the first phase, TFRP selects two fixed reference points, R_1 and R_2, which are two extreme points calculated from the micro-data set. Let a micro-data set contain p numerical attributes and n records. The two extreme values, *GMin* and *GMax,* are defined as follows:

$GMin = \min\{x_j(i) \mid 1 \le j \le n \text{ and } 1 \le i \le p\}$, and

$GMax = \max\{x_j(i) \mid 1 \le j \le n \text{ and } 1 \le i \le p\}$,

where $x_j(i)$ denotes the i-th attribute value for each vector x_j in the micro-data set. That is, *GMin* is the minimum attribute value over all micro-data set. Similarly, *GMax* is the maximum attribute value over all micro-data set. The p attribute values of the two reference points become

$R_1 = \{GMin, GMin, \ldots, GMin\}$, and

$R_2 = \{GMax, GMax, \ldots, GMax\}$.

If $GMin = GMax$, than $R_1 = R_2$. To avoid this situation, the record number of a micro-data set must be greater than one. After generating the two reference points, we can employ several techniques to reduce the running time of TFRP.

Example 1. Consider the nine two-dimensional vectors in Fig. 1. Each vector has a literal label and a number pair to show its position in the two-dimensional coordinate system. According to the above definition, $GMin = \min\{11, 9, 11, 8, 12, 6, 9, 6, 8, 10, 5, 4, 4, 3, 2, 5, 1, 3\} = 1$ and $GMax = \max\{11, 9, 11, 8, 12, 6, 9, 6, 8, 10, 5, 4, 4, 3, 2, 5, 1, 3\} = 12$. Therefore, $R_1 = \{1, 1\}$ and $R_2 = \{12, 12\}$.

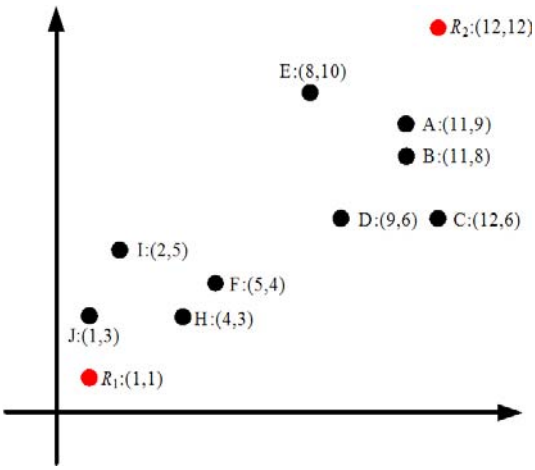


Fig. 1. Example of micro-data

3.1. Reduce the running time

In the first phase, TFRP is a fixed-size microaggregation method and requires $\left\lfloor \frac{n}{k} \right\rfloor$ iterations to generate $\left\lfloor \frac{n}{k} \right\rfloor$ groups. In each iteration, TFRP selects the *initial point* x_r, the furthest vector from a reference point, then computes the distance of each vector to x_r. Unlike the Diameter or the Centroid methods, which select the closest vector to the centroid of G_r, TFRP selects the k-1 closest vectors to x_r and x_r itself to form group G_r. After $\left\lfloor \frac{n}{k} \right\rfloor$ iterations, each remaining (n mod k) vector is added to its nearest group.

Since the two reference points R_1 and R_2 are fixed, to efficiently obtain each initial vector, this study employs two one-dimensional arrays to pre-compute, sort, and store the distances of R_1 and R_2 to all vectors. Furthermore, to obtain the k-1 closest vectors to the initial vector efficiently, the max-priority queue or the median-of-medians techniques [6] can be employed to speed up the process. The time complexity of the median-of-medians algorithm is linear; thus, this study implements the selection algorithm to TFRP. The two techniques cannot be applied to the Diameter or Centroid methods, because they must re-compute the centroid of a group while a new element is added. The pseudo-code of the first phase of TFRP is provided in Appendix A.

The algorithm of the first phase of TFRP is as follows:

Algorithm 1: Phase I of TFRP

1. Compute the two reference points R_1 and R_2. All vectors are assigned to a set (*SET*).
2. Select a reference point.
3. Select an initial point x_i from the reference point.
4. Calculate the distance of each vector to x_i.
5. Select k-1 closest vectors together with x_i to form a group, and remove the k vectors from *SET*.
6. Select another reference point, then go to Step 2 until $|SET|<k$.
7. Assign each remaining vector of *SET* to its closest group.

In Step 7, each remaining vector is assigned to its closest group. To avoid further unnecessary calculation of distances, this study applies the partial distance search (PDS) technique [4] for early termination of the computation of the squared Euclidean distance between the centroid of each group and a vector. Let the m-dimensional partial distance between x and y be $d^m(x, y) = \sum_{i=1}^{m}(x_i - y_i)^2$, where $1 \le m \le p$. Assuming that the smallest distance found so far to the vector x is d_{min}', if the centroid c_i of a group satisfies the condition $d^m(x, c_i) > d_{min}'$, then group G_i must be not the closest group to x. Accordingly, the process can reject G_i without calculating the real distance between x and c_i.

Theorem 1. Let n be the number of records, k be the minimum size of groups, and the attribute number p be a constant. The time complexity of Phase I of TFRP is $O(\frac{n^2}{k})$.

The proof of Theorem 1 is provided in Appendix B. Example 2 presents a simple example of how to work on Phase I of TFRP. Based on the grouping result of Phase I, the second phase of TFRP employs an adjusted strategy to lower the information loss. The algorithm of the second phase appears in Section 3.2.

Example 2. Consider the sample micro-data set in Fig. 1 with k = 3. The first phase of TFRP is a fixed-size microaggregation. TFRP first generates the two reference points, R_1 and R_2. Then, TFRP assigns the furthest vector "A" from R_1 as an initial vector. Starting from vector "A", the closest two vectors are vector "B" and "C". Therefore, we get the first group, G_1 = {A, B, C}. Next, TFRP selects the vector "J," which is the furthest vector from another reference point, R_2, to form the second group, G_2 = {J, H,

I}. TFRP employs the two reference points in turn to get the initial vector of each group. The last group, G_3, is {E, D, F} as shown in Fig. 2. The first element of each group is the initial vector. If the number of the remaining vectors is less than k, then each vector is assigned to a corresponding group where the group's centroid is closest to the vector.

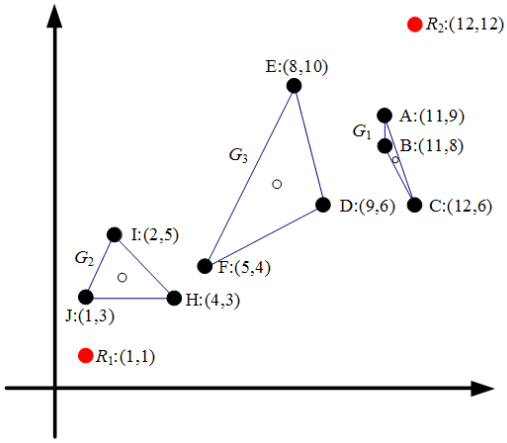


Fig. 2. Example of grouping micro-data in Phase I of TFRP

3.2. Decrease the information loss

After Phase I, TFRP generates $\left\lfloor \frac{n}{k} \right\rfloor$ groups with size $\geq k$. However, groups with size k do not always have the lowest SSE value. Several groups may have a large GSE value. Reassigning the members of a group with a large GSE value to the nearest groups may reduce the total SSE value. Therefore, to reduce the information loss in the second phase, TFRP sorts groups in decreasing order of their GSE values. Then, TFRP selects the groups in order and to check whether reassigning the members of the selected group to their nearest groups can improve the data quality. If scattering the selected group cannot reduce the information loss, then TFRP checks the next group; otherwise, TFRP scatters the group. The process terminates after each group is checked. The pseudo-code of the second phase of TFRP is presented in Appendix C.

The algorithm of the second phase of TFRP is as follows:

Algorithm 2: Phase II of TFRP

1. Compute GSE of each group, and sort them in decreasing order.

2. Select a group G_i in order and compute the current total sum of the within-group squared errors (SSE_1).
3. Calculate the distance of each vector of G_i to any other group.
4. Assign each vector of G_i to its closest group provisionally, and compute the current total sum of the within-group squared errors (SSE_2).
5. If $SSE_1 > SSE_2$, then assign each vector of G_i to its closest group; otherwise, regain G_i.
6. Return to Step 2 and repeat until each group is checked.

In Step 4, TFRP only calculates the GSE values of influenced groups to avoid redundantly computing the GSE values of changeless groups. The pseudo-code of Step 4, Compare_SSE(), is provided in Appendix C. In Step 5, scattering a group can reduce the information loss in each iteration. The process of Phase II reduces the number of groups to lower the information loss. Several groups may contain $2k$ or more members. Any fixed-size microaggregation method can be used to further partition oversized groups and reduce information loss. In Phase II, this study employs Phase I to further partition. To use the Phase I process in Phase II efficiently, this study constrains the size of oversized groups to be less than $4k$. If the size of the closest group to vector x has achieved $4k$-1, then the second closest group to x will be selected.

Theorem 2. Let n be the number of records, k be the minimum size of groups, and the attribute number p be a constant. The time complexity of Phase II of TFRP is $O(\frac{n^2}{k}+kn)$.

The proof of Theorem 2 is provided in Appendix D.

Example 3. Consider the same micro-data set as in Example 2 with k = 3. After Phase I, TFRP generates three groups G_1 = {A, B, C}, G_2 = {J, H, I} and G_3 = {E, D, F} as shown in Fig. 2. In Phase II, TFRP computes the *GSE* values of the three groups to obtain $GSE(G_1)$ = 5.33, $GSE(G_2)$ = 7.33 and $GSE(G_3)$ = 27.33, respectively. Therefore, $SSE_1 = GSE(G_1) + GSE(G_2) + GSE(G_3) = 40$. TFRP checks G_3, G_2 and G_1 (decreasing order of their GSE values) to see whether reassigning the members of a selected group to their nearest groups can lower the information loss. For G_3, the member vectors “D”, “E” and “F” is closest to groups G_1, G_1 and G_2, respectively. If

these vectors are reassigned to G_1 and G_2 to obtain two larger groups G_1' and G_2', respectively, then we get $GSE(G_1') + GSE(G_2') = 23.6 + 12.75 < 40$. As shown in Fig. 3, TFRP scatters G_3 to reach a lower information loss. Since breaking G_2' or G_1' cannot acquire a better result, the process is terminated.

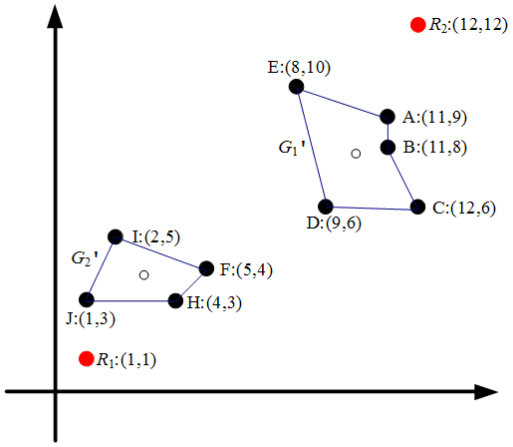


Fig 3. Example of re-grouping micro-data in Phase II of TFRP

The re-grouping technique of Phase II is a post-processing algorithm, which can be applied to any fixed-size microaggregation method. This study integrates the post-processing algorithm to the Diameter and Centroid methods and renames them Diameter+ and Centroid+, respectively. Thus, Diameter+ and Centroid+ are also two-phase methods.

Theorem 3. Let n be the number of records, k be the minimum size of groups, and the attribute number p be a constant. If $n>>k$, then the time complexity of the TFRP algorithm is $O(\frac{n^2}{k})$.

Proof. The TFRP algorithm is constituted by Phase I and Phase II. According to Theorems 1 and 2, the time complexity of TFRP is $O(\frac{n^2}{k}+\frac{n^2}{k}+kn) = O(\frac{n^2}{k})$ since $n>>k$.

Q.E.D

4. Experimental Results

This simulation used a 3.4 GHz Intel Pentium IV PC with 512 MB of main memory, running the Windows XP Professional operating system, to compare the performances of the Diameter, Diameter+, Centroid, Centroid+, and TFRP methods. To highlight the different contributions between Phase I and Phase II of TFRP, this experiment also implemented TFRP_I, which only included Phase I of TFRP. All algorithms were coded in Visual C++ 6.0 and applied to process three real micro-data sets and two images. To ensure the same unit of measurement for each attribute, all attributes were standardized in advance. Let $x(i)$ be the value of the i-th attribute $Attr_i$ of an arbitrary record x in a micro-data set. The standardization replaces each $x(i)$ with $\frac{x(i)-\overline{Attr_i}}{SD_i}$, where $\overline{Attr_i}$ and SD_i are the average and the standard deviation values of $Attr_i$, respectively, before performing the microaggregation process.

4.1. Datasets

This experiment used three real micro-data sets (Tarragona, Census and EIA) that have become usual reference datasets for testing multivariate microaggregation [5, 9, 15, 20]. The Tarragona dataset contains 834 companies' figures in the Tarragona area in 1995. Census and EIA were obtained from the Data Extraction System of the U.S. Bureau of the Census and the U.S. Energy Information Authority, respectively [5].

In Section 4.2, for a certain k, TFRP seems more suitable for a sparse micro-data set than for a dense one. To demonstrate the trend of TFRP, this experiment employs two image data to simulate two sets of dense and sparse datasets. Users can divide an image into several small blocks of an identical size. A block of the image data can be regarded as a record of the micro-data set. Each pixel value of the block becomes one of the attribute values of the record. Thus, users can divide an image into several blocks to easily obtain a dense or sparse dataset by tuning the block size. This experiment employed two famous images, Lena and Girl, of 512×512 pixels with 256 gray levels (as shown in Fig. 4) to generate several datasets. Each image was divided into various 2×2 and 4×4 blocks to obtain 4-dimensional dense and 16-dimensional sparse datasets, respectively. To investigate scalability with respect to dataset size,

this simulation united two 4-dimensional datasets to create a large dataset named Lena-Girl_2X2. Table 1 lists the summary of the datasets used in this experiment.

(a) Lena (b) Girl

Fig. 4. Images used

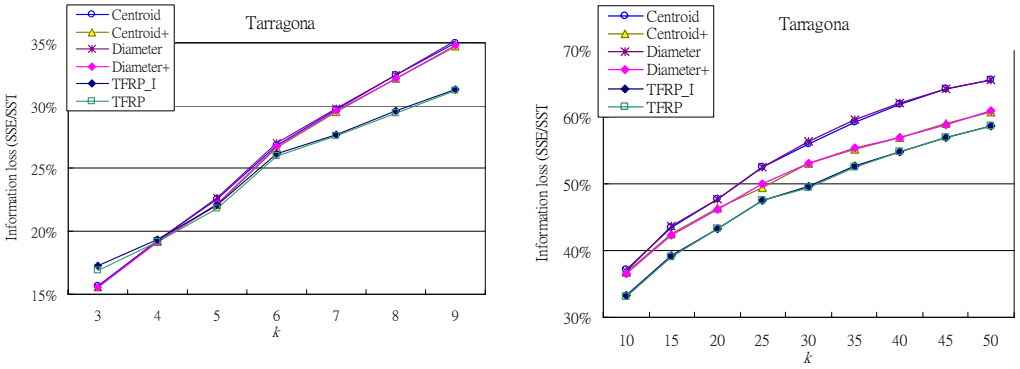
Table 1. Characteristics of experimental datasets

Dataset	Record number	Dimension
Tarragona	834	13
Census	1080	13
EIA	4092	11
Lena_2X2	65,535	4
Lena_4X4	16,384	16
Girl_2X2	65,535	4
Girl_4X4	16,384	16
Lena-Girl_2X2	131,070	4

4.2. Information loss comparison

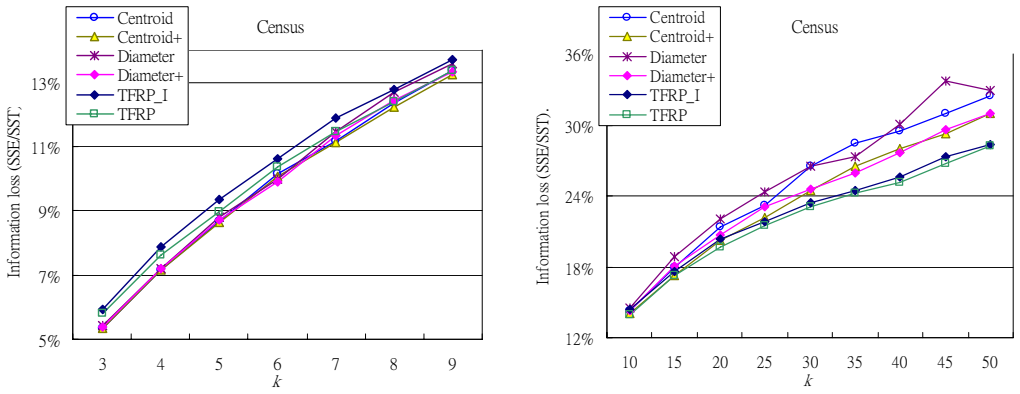
Figures 5 through 7 plot the performance curves of information loss over various k values with the six algorithms applied to Tarragona, Census, and EIA, respectively. The performances of Diameter+, Centroid+, and TFRP always overcame that of Diameter, Centroid, and TFRP_I, respectively. The post-processing technique can reduce the information loss significantly for a large k value. In Fig. 5, TFRP_I had the highest information loss when $k < 5$; TFRP performed best except where $k = 3$. In Fig. 6, TFRP also had the lowest information loss when $k \geq 10$. For example, when $k = 45$, TFRP had an information loss lower than the Diameter, Diameter+, Centroid, Centroid+, and TFRP_I methods by 6.94%, 2.86%, 4.29%, 2.49%, and 0.57%, respectively. The EIA dataset is known to be naturally clustered for $k = 6$; therefore,

the Diameter and Centroid methods performed better than TFRP. However, for a small k value, such as k = 5 or k = 7, TFRP significantly outperformed the Diameter and Centroid methods. In eight different k value scenarios (k = 5, 7, 9, 20, 35, 40, 45, or 50), TFRP had the lowest information loss as shown in Fig. 7. For a small k value, such as k = 3 to k = 6, TFRP_I performed the worst.



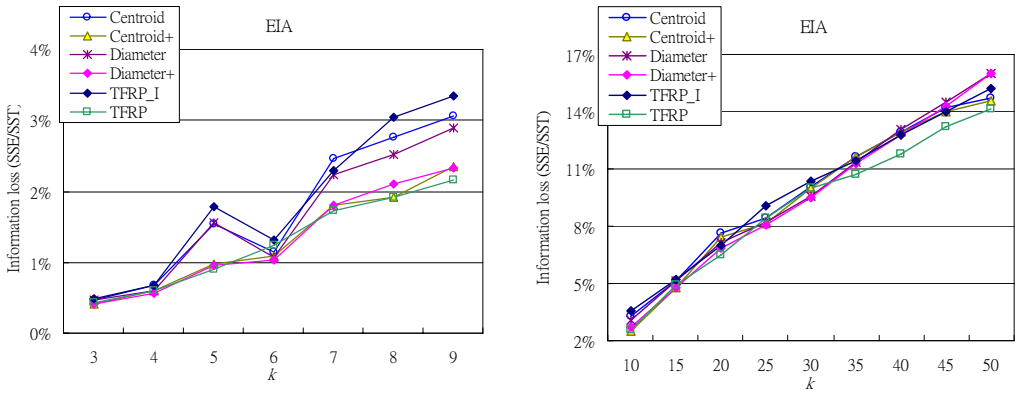
(a) k value between 3 and 9 (b) k value between 10 and 50

Fig. 5. Information loss comparison using Tarragona



(a) k value between 3 and 9 (b) k value between 10 and 50

Fig. 6. Information loss comparison using Census



(a) *k* value between 3 and 9 (b) *k* value between 10 and 50

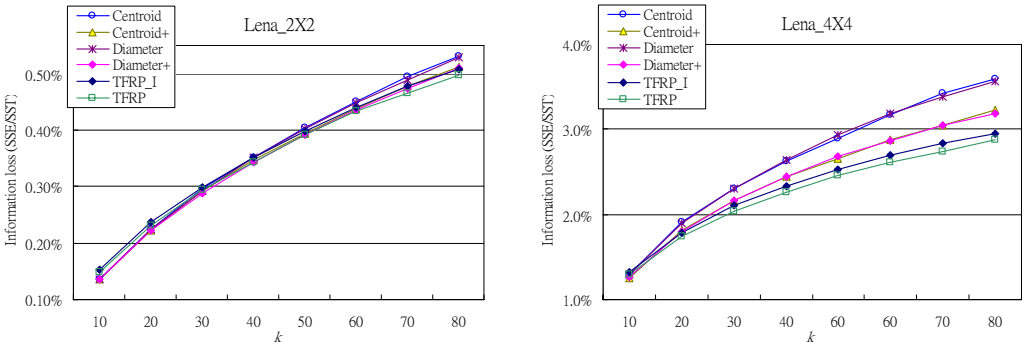
Fig. 7. Information loss comparison using EIA

Table 2 lists the information loss values for Phase I and Phase II of TFRP using Tarragona, Census, and EIA. Table 2 demonstrates that the second phase of TFRP always improved the data quality from the first phase. For the two sparse datasets, Tarragona and Census, the improvement was not very significant. These two datasets have no noticeable natural clusters for a certain *k*. The EIA dataset is naturally clustered for $k = 6$. Phase II reduced the information loss by only 0.18% from Phase I for $k = 6$. For $k = 5$, 7, 8, or 9, the data quality refinement was significant.

The data distribution in Tarragona is more sparse than that in the Census. In Figs. 5 and 6, when the *k* value increased, there was a greater decrease in information loss using TFRP compared to the other methods. Furthermore, for a certain *k*, TFRP was more suitable for a sparse micro-data set than for a dense one. To demonstrate these two trends of TFRP, this experiment also performed algorithms on two pairs of datasets: Lena_2X2, Lena_4X4 and Girl_2X2, Girl_4X4. Similar results were obtained, as shown in Figs. 8 and 9. In Fig. 8a, when $k \geq 60$, TFRP was the best algorithm; for $k = 10$, Centroid+ overcame the other methods. For a sparse dataset, TFRP had the lowest information loss when $k \geq 20$ as shown in Fig. 8b.

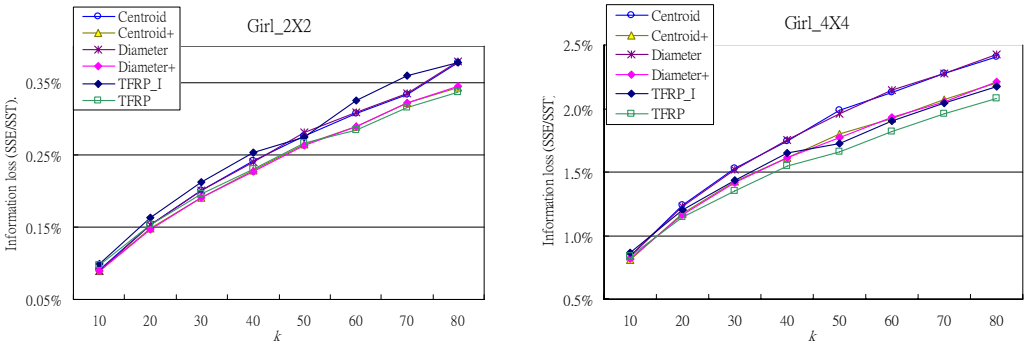
Table 2. Comparison of information loss (%) between Phase I and Phase II of TFRP

k	Tarragona		Census		EIA	
	Phase I (TFRP_I)	Phase I+II (TFRP)	Phase I (TFRP_I)	Phase I+II (TFRP)	Phase I (TFRP_I)	Phase I+II (TFRP)
3	17.228	16.881	5.931	5.803	0.530	0.428
4	19.396	19.181	7.880	7.638	0.661	0.599
5	22.110	21.847	9.357	8.980	1.651	0.910
6	26.220	25.971	10.623	10.357	1.416	1.238
7	27.695	27.636	11.874	11.476	2.348	1.728
8	29.625	29.441	12.775	12.411	2.729	1.920
9	31.303	31.247	13.699	13.360	2.959	2.151
10	33.186	33.088	14.442	13.959	3.242	2.590
15	39.166	39.120	17.606	17.216	5.198	4.922
20	43.315	43.264	20.289	19.629	6.567	6.518
25	47.551	47.438	21.795	21.460	8.472	8.443
30	49.554	49.466	23.474	23.068	10.202	10.015
35	52.693	52.590	24.474	24.184	11.416	10.710
40	54.809	54.731	25.638	25.188	11.802	11.761
45	56.880	56.867	27.291	26.721	13.224	13.194
50	58.597	58.568	28.310	28.224	14.171	14.122



(a) Dense dataset: Lena_2X2 (b) Sparse dataset: Lena_4X4

Fig. 8. Information loss comparison using Lena



(a) Dense dataset: Girl_2X2 (b) Sparse dataset: Girl_4X4

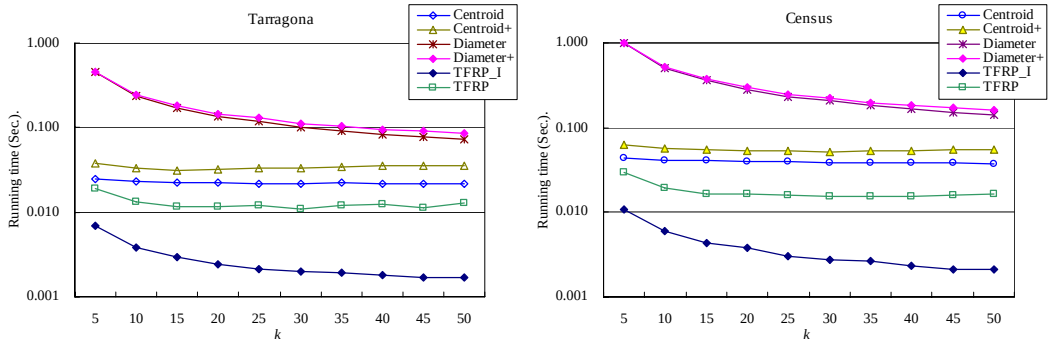
Fig. 9. Information loss comparison using Girl

4.3. Running time comparison

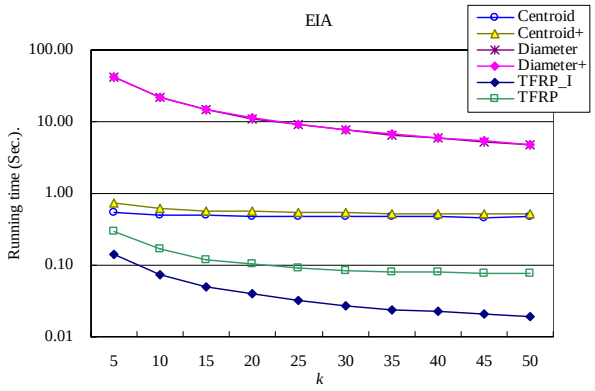
Each algorithm in this experiment had the same data reading process, which read the input micro-data set from disk. Therefore, to attain the running time of the three algorithms on different datasets and to eliminate the effect of disk I/O, the running time in this experiment did not include the cost to read the micro-data set from the disk.

Figures 10a to 10c illustrate the running time curves over various k values with the six algorithms applied to Tarragona, Census, and EIA, respectively. Each figure employs a logarithmic scale on its y-axis. The TFRP_I algorithm always performed the fastest, followed by TFRP. Centroid+ and Diameter+ performed slightly slower than Centroid and Diameter, respectively. The running time of the second phase is slight; thus, the second phase, which generally reduces the information loss, is considered as a cost-effective process. The running times of the Diameter, Centroid, and TFRP_I algorithms were consistent with the time complexity of $O(\frac{n^3}{k})$, $O(n^2)$, and $O(\frac{n^2}{k})$, respectively. In Fig. 10, for $k \geq 25$, increasing the k value for TFRP resulted in almost no running time improvement. This seems to violate the time complexity $O(\frac{n^2}{k})$. In fact, TFRP performs too fast for small datasets resulting in the Phase I process of TFRP do not dominate the overall performance. For a small dataset, the time complexity of TFRP becomes $O(\frac{n^2}{k}+kn)$ according to Theorem 2. For a large dataset, the portion of time complexity, $O(kn)$, can be neglected, as shown in Figs. 11 and 12.

Figures 11 and 12 demonstrate that TFRP performed faster than the Diameter method by more than three orders of magnitude. TFRP overcame the Centroid method, especially in a large k scenario. For example, in Fig. 11b where k = 10, the execution time of Centroid was 3.52 times that of TFRP. Along with growth of the k value, the difference increased to reach 12.84 times when k = 80. The running time curves of Diameter and Diameter+ (or Centroid and Centroid+) nearly overlap, because the second phase required less than 13 seconds to run.

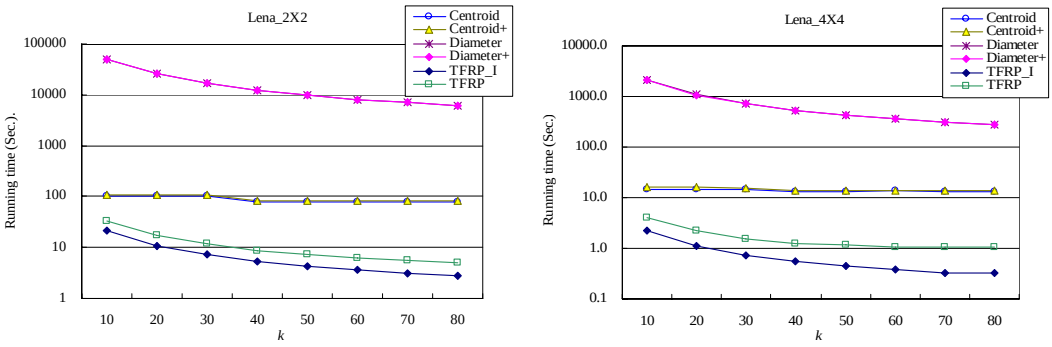


(a) Tarragona (b) Census



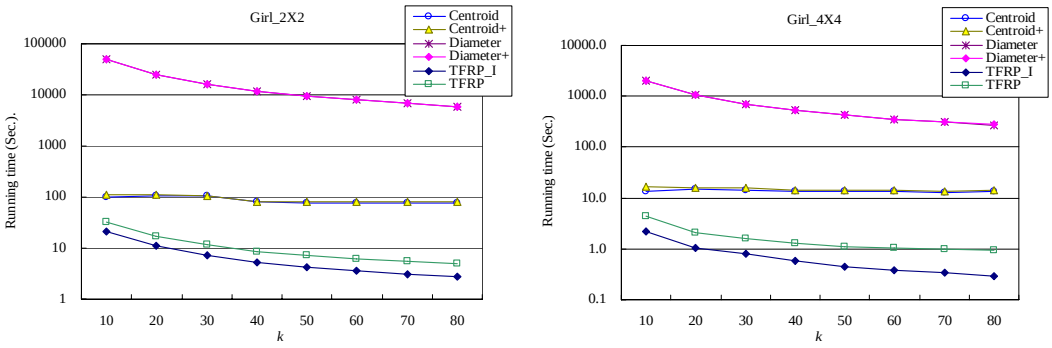
(c) EIA

Fig. 10. Running time comparison using three real datasets



(a) Lena_2X2 (b) Lena_4X4

Fig. 11. Running time comparison using Lena



(a) Girl_2X2 (b) Girl_4X4

Fig. 12. Running time comparison using Girl

To investigate the impact of dataset size on the running time of these six algorithms, this experiment executed algorithms on the Lena-Girl_2X2 dataset over various sizes between 10,000 and 120,000 for k = 50. As shown in Fig. 13, the Diameter method exhibited the cubic time behavior. For a certain k, although the running times of TFRP and Centroid increased quadratically with the growth of the dataset size, TFRP performed faster than Centroid by one order of magnitude.

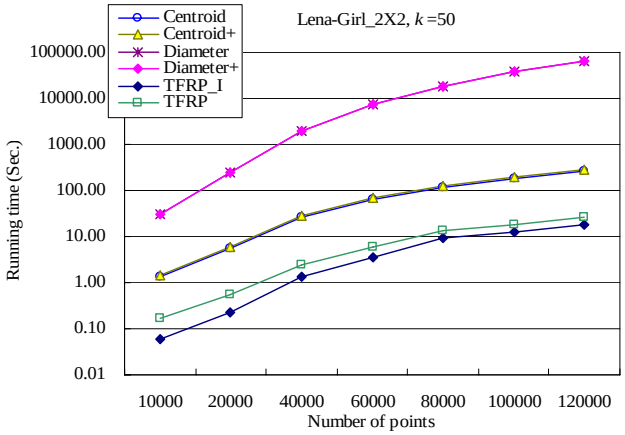


Fig. 13. Scalability comparison using Lena-Girl_2X2

The goal of the first phase of TFRP mainly focuses on speeding up the microaggregation process. Therefore, TFRP_I performed faster than all other methods as shown in Figs. 10 through 13. Although the information loss of TFRP_I was less than the Diameter and Centroid methods in Figs. 5b and 6b, TFRP_I still performed the worst in many cases, especially in Figs. 7 and 9a. For example, in Fig. 7a with k =

9, TFRF_I had the highest information loss: 3.35% greater than Centroid, Diameter, Centroid+, Diameter+, and TFRP by 0.28%, 0.45%, 1.01%, 1.03%, and 1.20%, respectively. Therefore, the second phase is still necessary for improving the data quality.

5. Conclusions

Statistic Disclosure Control (SDC) aims to release datasets without disclosing individual information of high security. Microaggregation is a useful technique for preserving the confidentiality of individual data. This study proposes a two-phase method, Two Fixed Reference Points (TFRP), for microaggregation. In the first phase, TFRP employs efficient techniques, such as median-of-medians and PDS, to speed up the microaggregation process. The time complexity using TFRP is reduced to become $O(\frac{n^2}{k})$ for a large dataset. Furthermore, in the second phase, TFRP regroups several groups by removing large GSE groups in order to the reduce information loss. The post-processing technique of the second phase can be applied to any fixed-size microaggregation method.

Simulation results reveal that the running time of the proposed method outperforms that of the Diameter and Centroid methods. TFRP also reduces the information loss significantly, particularly in sparse datasets with a large k value. Although TFRP is fast, the time complexity of TFRP still reaches $O(n^2)$ for a certain k. In the future, the authors will consider the development of superior algorithms that can efficiently support a huge database.

Appendix A (pseudo-code of Phase I)

The pseudo-code of Phase I of TFRP is as follows:

Input: (1) *SET*: A micro-data set with *p* attributes and *n* data vectors

(2) *k*: minimum group size

Output: *GSET*: $\left\lfloor \frac{n}{k} \right\rfloor$ groups with size $\geq k$

Procedure: Phase I

```
01. Compute the two references R1 and R2;
02. ∀x∈SET, compute d(x, R1) and sort them in decreasing order
    on array Dis[1][];
03. ∀x∈SET, compute d(x, R2) and sort them in decreasing order
    on array Dis[0][];
04. GSET := ∅;
05. for(i:=1 to ⌊n/k⌋) {
06.   nSwitch := i%2;
07.   Select the initial vector xr from Dis[nSwitch][];
08.   Gi := xr;
09.   SET := SET-xr;
10.   ∀x∈SET {
11.         calculate d(x, xr); }
12.   x' = kth_element(SET);
13.   ∀x∈SET {
14.         if x.distance ≤ x'.distance {
15.               Gi := Gi+x;
16.               SET := SET-x;   }    }
17.   GSET := GSET+Gi; }
18. Assign(SET);
19. return GSET;
```

In Lines 6 and 7, the process totally scans two sorted one-dimensional arrays only once to obtain $\left\lfloor \frac{n}{k} \right\rfloor$ initial vectors. Line 12 employs median-of-medians to select the *k*th-smallest distance to the initial point to speed up the process of forming a group with size *k*. In Line 18, the Assign() function adds each of the remaining vectors to its closest group. The pseudo-codes of functions Assign() and PDS() are as follows:

```
Assign(SET) {
01.   ∀x∈SET {
02.         d_min := ∞; // the initial minimum distance
03.         ∀G_i∈GSET {
04.               PDS(x, G_i.centroid, d_min);
05.               if (d^m(x, G_i.centroid)<d_min) {
06.                     d_min := d^m(x, G_i.centroid);
07.                     x.closest := G_i; }    }
08.         G_i := G_i+x;
09.         SET := SET-x;   }
}

PDS(x, y, d_min) {
01.   for(m:=1 to p) {
02.         if (d^m(x, y) > d_min) {
03.               break; }   }
}
```

Appendix B (time complexity of Phase I)

Theorem 1. Let n be the number of records, k be the minimum size of groups, and the attribute number p be a constant. The time complexity of Phase I of TFRP is $O(\frac{n^2}{k})$.

Proof. To analyze the time complexity of Phase I of TFRP, Phase I is divided into four parts as follows:

(1) Lines 1-4: In Line 1 of Phase I, computing two reference points requires $O(2\times p\times n) = O(n)$. In Lines 2 and 3, Phase I calculates the distance of each vector to each reference point and sorts all distances. Thus, the time complexity of Lines 2 and 3 is $O(2\times p\times n + n\times \log n) = O(n\times \log n)$. The running time of Lines 2 and 3 dominates the total running time of Lines 1 to 4. That is, $O(n\times \log n)$.

(2) Lines 6-9: In Line 7, the process scans two sorted one-dimensional arrays once to obtain $\left\lfloor \frac{n}{k} \right\rfloor$ initial vectors. Thus, the running time is $O(2n) = O(n)$. Other lines have no higher running time.

(3) Lines 10-17: In Line 11, Phase I computes distance $\left\lfloor \frac{n}{k} \right\rfloor \times n$ times. The running time of Line 12 is $O(\left\lfloor \frac{n}{k} \right\rfloor \times n \times p) = O(\frac{n^2}{k})$. In Line 12, selecting the kth-smallest vector requires running time $O(n)$. The total running time of Line 12 become

$O(\lfloor \frac{n}{k} \rfloor \times n) = O(\frac{n^2}{k})$. In Lines 13 to 16, the running time of each iteration is $O(n)$. The time complexity of Lines 13 to 16 is $O(\lfloor \frac{n}{k} \rfloor \times n) = O(\frac{n^2}{k})$. Therefore, the expected time is $O(\frac{n^2}{k})$.

(4) Line 18: In the worst case, Assign() computes the distances of k-1 vectors to $\lfloor \frac{n}{k} \rfloor$ groups. Therefore, the expected time of Line 18 is $O(p \times (k-1) \times \lfloor \frac{n}{k} \rfloor) = O(n)$.

According to the four parts and $\frac{n^2}{k} >> n \times \log n$, the time complexity of Phase I is dominated by $O(\frac{n^2}{k})$.

Q.E.D

Appendix C (pseudo-code of Phase II)

The pseudo-code of Phase II of TFRP is as follows:

Input: *GSET*: The generated $\lfloor \frac{n}{k} \rfloor$ groups from Phase I with size $\geq k$

Output: *GSET'*: The re-grouping set with a lower information loss

Procedure: Phase II

```
01. ∀G∈GSET, compute GSE(G) and sort them in decreasing order;
02. for(i:=1 to ⌊n/k⌋) {
03.   if(G_i.size ≥ 2k) {
04.         continue;  }
05.   FakeAssign(G_i);
06.   diff := Compare_SSE(G_i);
07.   if(diff>0) {
08.         ∀x∈G_i assign x to a proper group;
09.         GSET := GSET-G_i; }     }
10. ∀G∈GSET, if(G_i.size ≥ 2k) { // G_i.size < 4k
11.   Execute the process of Phase I to partition group G_i; }
12. return GSET;
```

In Line 5, the function of FakeAssign() is similar to the function of Assign() in Phase I. Instead of assigning a vector to its closest group, FakeAssign() keeps each

vector of the selected group until the process executes Line 8. Without redundancy, this study omits the detailed pseudo-code of FakeAssign(). In Line 6, Compare_SSE() checks to see whether the current SSE value is greater than that of the SSE value after each member of G_i is reassigned to its closest group. The pseudo-code of Compare_SSE() is as follows:

```
Compare_SSE(Gi) {
// Let x∈Gi, x.closest := Gj, where i≠j and Gj∈GSET
// Initially, Gj'==Gj
// ∀x∈Gi, Gj' := Gj' + x, where x.closest:= Gj and Gj.size<4k
01. SSE1 = ∑GSE(Gj)+GSE(Gi);
02. ∀x∈Gi, compute GSE(Gj')
03. SSE2 = ∑GSE(Gj');
04. return SSE1-SSE2;
}
```

The function Compare_SSE() computes how much SSE value can be reduced after G_i is scattered. This function avoids re-computing the GSE values of groups where the GSE value has not changed. The size of each group is constrained less than $4k$ to efficiently partition the oversize groups.

Appendix D (time complexity of Phase II)

Theorem 2. Let n be the number of records, k be the minimum size of groups, and the attribute number p be a constant. The time complexity of Phase II of TFRP is $O(\frac{n^2}{k}+kn)$.

Proof. The running time of Phase II is determined by Lines 1, 5, 6 and 11. The discussion is as follows:

(1) Line 1: The running time of computing the GSE value of a group requires $O(p\times k)$. The process totally computes the GSE values of $\left\lfloor \frac{n}{k} \right\rfloor$ groups and sorts them. Therefore, the expected time of Line 1 is $O(\ p\times k\times\left\lfloor \frac{n}{k} \right\rfloor+\left\lfloor \frac{n}{k} \right\rfloor\times\log\left\lfloor \frac{n}{k} \right\rfloor\ )$ = $O(n+\frac{n}{k}\times\log\frac{n}{k})$.

(2) Line 5: The time complexity of FakeAssign() is equal to that of the Assign() function of Phase I. That is, $O(n)$. Since Phase II executes FakeAssign() $\left\lfloor \frac{n}{k} \right\rfloor$ times, the running time is $O(\left\lfloor \frac{n}{k} \right\rfloor \times n) = O(\frac{n^2}{k})$.

(3) Line 6: In the worst case, Phase II executes the Compare_SSE() function $O(\frac{n}{k})$ times. Compare_SSE() requires to compute the GSE values of $4k$-1 groups with size $4k$-1. It has to compute n distances at most. Therefore, the running time is $O(\frac{n}{k} \times p \times (4k)^2) = O(kn)$.

(4) Line 11: In the worst case, Phase II calls Phase I $O(\frac{n}{k})$ times. Furthermore, the size of any oversized group is less than $4k$. The running time of Line 11 becomes $O(\frac{n}{k} \times \frac{\log k}{k}(4k)^2) = O(n \times \log k)$.

According to the four cases, with $\frac{n^2}{k} >> n + \frac{n}{k} \times \log \frac{n}{k}$ and $\frac{n^2}{k} >> n \times \log k$, the time complexity of Phase II is $O(\frac{n^2}{k} + kn)$.

Q.E.D

Acknowledgements

The authors would like to acknowledge the helpful comments made by the anonymous reviewers of this paper.

References

[1] N.R. Adam and J.C. Wortmann, "Security-control methods for statistical databases: a comparative study," *ACM Computing Surveys*, vol. 21, no. 4, pp. 515-556, 1989.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, "Approximation algorithm for *k*-anonymity," *Journal of Privacy Technology*, 2005.

[3] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios, "Disclosure limitation of sensitive rules," in *Proc. 1999 Workshop on Knowledge and Data Engineering Exchange*, Chicage, IL, pp. 45-52, November 1999.

[4] C.-D. Bei and R.M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. Communications*, vol. 33, no. 10, pp. 1132-1133, 1985.

[5] R. Brand, J. Domingo-Ferrer, and J.M. Mateo-Sanz, "Reference data sets to test and compare sdc methods for protection of numerical microdata," *European Project IST-2000-25069 CASC*, 2002. Available at: http://noen.vb.cbs.nl/casc (accessed June 28, 2006).

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, "Introduction to algorithms," *MIT Press*, 2nd Edition, Cambridge, MA, 2003.

[7] J. Domingo-Ferrer, "Advances in inference control in statistical database: an overview," in J. Domingo-Ferrer and V. Torra (Eds.), *Lecture Notes in Computer Sciences* 3050 --- *CASC Project Intl. Workshop on Privacy in Statistical Database* (*PSD 2004*), Springer-Verlag, Berlin, pp. 1-7, 2004.

[8] J. Domingo-Ferrer, "Microaggregation: Achieving *k*-anonymity with quasi-optimal data quality," in *Proc. 2006 European Conference on Quality in Survey Statistics*, Cardiff, UK, April 2006.

[9] J. Domingo-Ferrer and J.M. Mateo-Sanz, "Practical data-oriented microaggregation for statistical disclosure control," *IEEE Trans. Knowledge and Data Engineering*, vol. 14, no. 1, pp. 189-201, 2002.

[10] J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogenerous *k*-anonymity through microaggregation," *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 195-202, 2005.

[11] G.T. Ducan and S. Mukherjee, "Optimal disclosure limitation strategy in statistical databases: deterring tracker attacks through additive noise," *Journal of the American Statistical Association*, vol. 95, no. 451, pp. 720-729, 2000.

[12] S.L. Hansen and S. Mukherjee, "A polynomial algorithm for optimal univariate microaggregation," *IEEE Trans. Knowledge and Data Engineering*, vol. 15, no. 4, pp. 1043-1044, 2003.

[13] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering: a review," *ACM Computer Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[14] M. Kantardzic, "Data mining: concepts, models, methods, and algorithms," *John Wiley & Sons*, New York, 2002.

[15] M. Laszlo and S. Mukherjee, "Minimum spanning tree partitioning algorithm for microaggregation," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 7, pp. 902-911, 2005.

[16] C.K. Liew, U.J. Choi, and C.J. Liew, "A data distortion by probability distribution," *ACM Trans. Database Systems*, vol. 10, no. 3, pp. 395-411, 1985.

[17] A. Meyerson and R. Williams, "On the complexity of optimal *k*-anonymity," in *Proc. 23rd ACM Symposium on Principles of Database Systems*, Paris, France, pp. 223-228, June 2004.

[18] A. Oganian and J. Domingo-Ferrer, "On the complexity of optimal microaggregation for statistical disclosure control," *Statistical Journal of United Nations Economic Commission for Europe*, vol. 18, no. 4, pp. 345-354, 2001.

[19] P. Samarati, "Protecting resopndents' identities in microdata release," *IEEE Trans. Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010-1027, 2001.

[20] A. Solanas, A. Martínez-Ballesté, J. Domingo-Ferrer, and J.M. Mateo-Sanz, "A 2^d-tree-based blocking method for microaggregating very large data sets," in *Proc. 1st Intl. Conf. on Availability, Reliability and Security*, Los Alamitos, CA, pp. 922-928, April 2006.

[21] L. Sweeney, "*k*-Anonymity: a model for protecting privacy," *Intl. Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557-570, 2002.

[22] J.F. Traub, Y. Yemini, and H. Wozniakowski, “The statistical security of a statistical database,” ACM Trans. on Database Systems, vol. 9., no. 4, pp. 672-679, 1984.

[23] V. Torra, “Microaggregation for categorical variables: A median based approach,” in J. Domingo-Ferrer and V. Torra (Eds.), *Lecture Notes in Computer Sciences* 3050 --- *CASC Project Intl. Workshop on Privacy in Statistical Database* (*PSD 2004*), Springer-Verlag, Berlin, pp. 162-174, 2004.

[24] J.H. Ward, Jr., “Hierarchical grouping to optimize an objective function,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236-244, 1963.