The ISC Int'l Journal of Information Security

January 2020, Volume 12, Number 1 (pp. 35–43)

http://www.isecure-journal.org

Improved Univariate Microaggregation for Integer Values

Reza Mortazavi [1,*]

[1] *Computer Engineering Department, School of Engineering, Damghan University, Damghan, Iran*

ARTICLE INFO.

Article history:
Received: 12 May 2019
Revised: 26 August 2019
Accepted: 10 October 2019
Published Online: 22 December 2019

Keywords:
Data Privacy, Statistical Databases, Microdata Protection, Microaggregation, Integer Optimization.

Abstract

Privacy issues during data publishing is an increasing concern of involved entities. The problem is addressed in the field of statistical disclosure control with the aim of producing protected datasets that are also useful for interested end users such as government agencies and research communities. The problem of producing useful protected datasets is addressed in multiple computational privacy models such as k-anonymity in which data is clustered into groups of at least k members. Microaggregation is a mechanism to realize k-anonymity. The objective is to assign records of a dataset to clusters and replace the original values with their associated cluster centers which are the average of assigned values to minimize information loss in terms of the sum of within group squared errors (SSE). While the problem is shown to be NP-hard in general, there is an optimal polynomial-time algorithm for univariate datasets. This paper shows that the assignment of the univariate microaggregation algorithm cannot produce optimal partitions for integer observations where the computed centroids have to be integer values. In other words, the integrality constraint on published quantities has to be addressed within the algorithm steps and the optimal partition cannot be attained using only the results of the general solution. Then, an effective method that considers the constraint is proposed and analyzed which can handle very large numerical volumes. Experimental evaluations confirm that the developed algorithm not only produces more useful datasets but also is more efficient in comparison with the general optimal univariate algorithm.

1 Introduction

Considering the confidentiality of individuals and enterprises is a vital part of microdata publishing. The Statistical Disclosure Control (SDC) [1] aims to balance the privacy and utility of published data. The problem is addressed in multiple computational privacy models such as k-anonymity [2] in which data is partitioned into groups of at least k records. Microaggregation is a mechanism to implement k-anonymity in the SDC [3]. The method is currently being used by many statistical agencies [4]. Microaggregation can be considered as a constraint-clustering problem, where records have to be assigned to groups of a specified minimum size. After partitioning records into clusters, they are replaced by their associated centroids to mask the detailed information of records. Therefore, the attacker cannot limit a record in a small group. The method is introduced for continuous numerical attributes, while there are extensions

* Corresponding author.
Email address: r_mortazavi@du.ac.ir
ISSN: 2008-2045

for other data types [5, 6], and even more complex data structures [7, 8] or privacy requirements [9].

The potency of a microaggregation algorithm is measured in terms of its incurred information loss which is measured by the sum of the squared distances between original records and their replaced values, SSE. It is proved that microaggregation is NP-Hard for multivariate datasets [10], while there exists a polynomial-time optimal algorithm called MHM for the univariate case [11]. In some applications, the computed centroids have to be integers. For instance, in k-degree anonymization of a network, all vertices have to meet k-anonymity for their degrees [12]. Casas-Roma et al. used the assignment of the MHM on ordered degree sequence of the original graph to achieve k-degree anonymity [13]. The authors used the floor and ceiling functions to round the computed centroids of the MHM as the degree of vertices in each group.

This study shows that the naïve application of the assignment of records to clusters produced by the MHM is not necessarily optimal for integer values. Inspired from the incremental computation introduced in the improved MHM called IMHM [14], an Improved univariate microaggregation algorithm based on MHM for Integers, IMHMI is developed. The proposed method produces more useful protected datasets in terms of SSE and is also more efficient than the MHM.

Section 2 reviews some basic concepts used in the study and Section 3 describes the new algorithm. Experimental results are given in Section 4, and finally, Section 5 concludes the paper.

2 Basic Concepts

In this section, the univariate microaggregation problem and its extensions for multivariate datasets are described. Section 2.1 formalizes the univariate microaggregation problem for integer observations. Section 2.2 reviews some heuristics to utilize the univariate algorithm for multivariate datasets.

2.1 The Univariate Microaggregation Problem of Integer Values

Let $X = (x_1, x_2, \ldots, x_n)$ denotes a vector of n integer observations from a sensitive attribute. A microaggregation algorithm partitions n values into groups such that each group has at least k observations. The sum of within-group squared errors SSE is shown in (1).

$$SSE = \sum_{j=1}^{g} \sum_{i=1}^{n_j} (x_{ij} - G_j)^2 \qquad (1)$$

The measure is usually used to evaluate the algorithm, where g shows the number of groups, x_{ij} is the ith number in the jth group, $i \in \{1, 2, \ldots, n_j\}, j \in \{1, 2, \ldots, g\}$ and G_j is the centroid of the jth group. The aim of the algorithm is a partition such that SSE is minimized. The partition can be shown using a vector of assignments $A = (a_1, \ldots, a_n)$ for $a_i \in \{1, 2, \ldots, g\}$ where x_i is assigned to the group number a_i. Generally, given a partition of numbers, it can be easily shown that the average of the values in each group minimizes SSE, i.e., $G_j = 1/n_j \sum_{i=1}^{n_j} x_{ij}$.

Domingo-Ferrer and Mateo-Sanz showed that in an optimal partition, each group has at most $2k - 1$ observations [15]. Hansen and Mukherjee developed a polynomial-time algorithm for optimal univariate microaggregation [11] called MHM in this study. The MHM first sorts data values and then constructs a directed acyclic graph in which each edge of the graph corresponds to a possible group that may be part of an optimal partition. The authors showed that the optimal microaggregation corresponds to computing the shortest path in the graph. A group exists in the optimal partition if its corresponding edge is included in the computed shortest path. The total complexity of the algorithm is $O(max(nlog(n), k^2n))$. Mortazavi and Jalili proposed an improved implementation of the MHM called IMHM that exploits incremental edge weight computations to reduce the complexity of graph construction to $O(kn)$ operations [14].

2.2 Multivariate Microaggregation Heuristics based on MHM

The optimal property of MHM provides a promising approach to solve the NP-hard problem of multivariate microaggregation. However, it is not well-defined how to sort multivariate datasets. Therefore, different heuristics are proposed in the literature to sequence multivariate records.

Domingo-Ferrer et al. [16] proposed some heuristics such as the Nearest Point Next (NPN-MHM), MDAV-MHM, and CBFS-MHM to order records and form a sequence of them. Then, MHM is applied to records on the path. Nin et al. used two projection methods, i.e., PCA and Z-score to reduce the dimension of the underlying dataset to one [17]. In the PCA technique, the first principal component of the dataset is used to sort data records [1]. The Z-score algorithm orders multivariate records based on the sum of their Z-scores. Additionally, Mortazavi and Jalili introduced the FDM that produces an optimal assignment of records with respect to a TSP tour of

[1] The technique is also shown to be effective in reducing the runtime of multivariate microaggregation [18].

them for a range of group sizes k [19]. Recently, Soria-Comas and Domingo-Ferrer presented a method to achieve differential privacy [20] through univariate microaggregation [9].

3 Proposed Method

In this section, the problem of univariate microaggregation of integer observations is solved using the shortest path problem, similar to [11]. Obviously, the nearest integer value to the average of a group of integer observations minimizes SSE for the group, i.e., $G_j = round(\sum_{i=1}^{n_j} x_{ij})$ [2]. However, it can be easily verified that the same assignment of records to clusters in the MHM does not necessarily result in the optimal partition for integers. For instance, consider $X = (1, 2, 2, 3, 4, 4, 6, 6, 8, 10)$ and $k = 3$. The MHM results in the assignment $A = (1, 1, 1, 2, 2, 2, 3, 3, 3, 3)$ with $G_1 = 1.67, G_2 = 3.67, G_3 = 7.5$ and $SSE = 12.33$. Using the rounded centroids, this assignment yields $G_1 = 2, G_2 = 4, G_3 = 8$, and $SSE = 14$. However, with this constraint, the assignment $A' = (1, 1, 1, 1, 2, 2, 2, 3, 3, 3)$ outputs $G'_1 = 2, G'_2 = 5, G'_3 = 8$ with $SSE' = 13$. This counterexample confirms that the trivial application of the assignment of the MHM or IMHM does not necessarily construct the optimal partition when the integer constraint exists. This is due to the way the centroids are computed for integer values during computing the shortest path. However, the same procedure of graph construction and partition in the MHM [11] can be adopted for integer values. In Section 3.1, the pseudo-code of the IMHMI is proposed and described, and in Section 3.2 it is analyzed.

3.1 The Pseudo-Code of the IMHMI

In this section, the main steps of the IMHMI are proposed. Algorithm 1 shows how IMHMI clusters a univariate dataset of n integer observations and assigns them to cluster centroids.

The algorithm accepts a vector of n integer observations $X = (x_1, x_2, \ldots, x_n)$, the privacy parameter k and outputs SSE and the assignment of observations to g clusters using $A = (a_1, a_2, \ldots, a_n)$, $a_i \in \{1, \ldots, g\}$ which means x_i is assigned to the group number a_i. Similar to MHM, all values of X are sorted and stored in the vector $V = (v_1, v_2, \ldots, v_n)$. The sort function also returns the sort index I such that $v_i = a_{I_i}$ (Line 1). Given V and k, a directed graph is constructed in which a node with label i represents the element v_i of V. The graph also has one additional dummy node v_0 with the value of $-\infty$

Algorithm 1: The pseudo-code of the IMHMI

Input: $X = (x_1, x_2, \ldots, x_n)$: vector of integer observations, k: the privacy parameter
Output: SSE: microaggregation error, $A = (a_1, a_2, \ldots, a_n)$: the assignment

1: $(V, I) \leftarrow sort(X)$ ▷ ascending sort
2: $v_0 \leftarrow -Inf$ ▷ inserts v_0 at the beginning of V
3: $parent \leftarrow Zero(0:n)$, $c \leftarrow 0$, $cost \leftarrow Inf(0:n)$, $cost_0 \leftarrow 0$
4: **for** $s \leftarrow 0 : n-k$ **do**
5: **if** $s == 0$ **then**
6: $sum \leftarrow \sum_{i=1}^{k-1} v_i$
7: **else**
8: $sum \leftarrow sum^* - v_s$
9: **for** $e \leftarrow s+k : \min(n, s+2k-1)$ **do**
10: $sum' \leftarrow sum$, $sum \leftarrow sum + v_e$, $c' \leftarrow c$
11: $c \leftarrow rDiv(sum, e-s)$
12: **if** $e == s+k$ **then**
13: **if** $e == k$ **then**
14: $SSE \leftarrow \sum_{i=1}^{k}(v_i - c)^2$ ▷ Equation (1)
15: **else**
16: $\Gamma = c - c^*$ ▷ Equation (4)
17: $SSE \leftarrow SSE^* + \Gamma(k\Gamma + 2(kc^* - sum^*)) + (v_e + v_s - 2c)(v_e - v_s)$
18: $c^* \leftarrow c$, $sum^* \leftarrow sum$, $SSE^* \leftarrow SSE$
19: **else**
20: **if** $cost_{e-k} < cost_s$ **then**
21: **break** ▷ terminates the inner loop
22: $\Delta = c - c'$ ▷ Equation (3)
23: $SSE \leftarrow SSE + (v_e - c)^2 - \Delta\,(2sum' - (e-s-1)(c'+c))$
24: **if** $cost_s + SSE < cost_e$ **then**
25: $cost_e \leftarrow cost_s + SSE$, $parent_e \leftarrow s$
26: $SSE \leftarrow cost_n$, $curNode \leftarrow n$, $g \leftarrow 0$
27: **while** $curNode \neq 0$ **do**
28: $newNode \leftarrow parent_{curNode}$
29: $a_{I_i} \leftarrow g \;\; \forall i = newNode + 1 : curNode$
30: $g \leftarrow g + 1$, $curNode \leftarrow newNode$
31: **return** SSE, $A = (a_1, a_2, \ldots, a_n)$

at the beginning of V (Line 2). For all consecutive group of nodes with labels $\{i, i+1, \ldots, j\}$ such that $i + k \leq j < i + 2k$, an arc (i, j) is added to the graph that corresponds a cluster $C_{(i,j)}$ of $\{v_{i+1}, \ldots, v_j\}$. Let $S_{(i,j)} = \sum_{h=i+1}^{j} v_h$. The length $L_{(i,j)}$ of the arc equals to SSE of $C_{(i,j)}$, i.e., $L_{(i,j)} = \sum_{h=i+1}^{j}(v_h - G_{(i,j)})^2$, where $G_{(i,j)} = round\left(S_{(i,j)}/(j-i)\right)$ denotes the centroid of $C_{(i,j)}$. The optimal partition is the set of groups that correspond to the arcs of the shortest path from node 0 to node n on the graph. The construction and computing the shortest path are simultaneously performed in Lines 3-25. In order to accelerate the construction process, the length of the arcs, except the first one, are computed incrementally. Line 3 initializes the required variables. Each arc of the graph starts from s and ends in e, which are the

[2] The method *round half away from zero* (or *round half towards infinity*) is used in this study. For example, round(7.5)=8 and round(-7.5)=-8.

loop indexes of the outer and inner loops, respectively. The (partial) sum of node values of the corresponding cluster, i.e., $\sum_{i=s+1}^{e-1}$ is computed in Lines 6-8. The value is stored in *sum* and is completed by adding v_e in Line 10. The function $rDiv$ calculates the integer centroids in Line 11. For $a \in \mathbb{Z}$ and $b \in \mathbb{Z}^+$, let $rDiv(a,b) = round(a/b)$. The function can be efficiently computed as shown in Equation (2).

$$rDiv(a,b) = \begin{cases} div(a + div(b,2), b) & \text{if } a \geq 0 \\ div(a - div(b,2), b) & \text{if } a < 0 \end{cases} \quad (2)$$

In Equation (2), div denotes integer division. The length of the first arc, i.e., $L_{(0,k)}$ is calculated in Line 14, but the following lengths are computed incrementally based on the previously calculated values (Lines 16-23). The following facts formalize the idea of incremental computations to construct the graph in the integer domain. [3]

(1) Let $\Delta = G_{(i,j+1)} - G_{(i,j)}$. The length $L_{(i,j+1)}$ of the arc $C_{(i,j+1)}$ can be computed efficiently as shown in Equation (3):

$$\begin{aligned} L_{(i,j+1)} = \\ L_{(i,j)} + (v_{j+1} - G_{(i,j+1)})^2 \\ - \Delta\Big(2S_{(i,j)} - (j-i)(G_{(i,j)} + G_{(i,j+1)})\Big). \end{aligned} \quad (3)$$

(2) Let $\Gamma = G_{(i+1,i+k+1)} - G_{(i,i+k)}$. The length $L_{(i+1,i+k+1)}$ of the arc $C_{(i+1,i+k+1)}$ can be computed efficiently as formulated in Equation (4):

$$\begin{aligned} L_{(i+1,i+k+1)} = \\ L_{(i,i+k)} + \Gamma\Big(k\Gamma + 2(kG_{(i,i+k)} - S_{(i,i+k)})\Big) \\ + \Big((v_{i+k+1} + v_{i+1} - 2G_{(i+1,i+k+1)}) \\ (v_{i+k+1} - v_{i+1})\Big). \end{aligned} \quad (4)$$

Moreover, the following fact lets the IMHMI ignore computation of the length of some arcs (Line 21), which improves the efficiency of computation of the shortest path. [4]

(1) For $k < j < 2k$ and $0 < i' < j$, we have $L_{i,i+j} \geq L_{i+i',i+j}$. Therefore, computation of $L_{i,i+j}$ can be skipped if there exists a node $i + i'$ with a shorter total length from node 0, since a path to $i+j$ will be shorter going through node $i+i'$ than the node i.

After the graph construction and finding the shortest path, the total SSE error equals to the length of the shortest path which is saved in SSE in Line 26. The assignment of observations to clusters is performed in Lines 28-30. Finally, SSE and A are returned in Line 31.

3.2 Analysis of the IMHMI Algorithm

The IMHMI consists of 4 main phases: (1) sorting data, (2) graph construction, (3) shortest path computation, (4) assigning records to groups. The first phase requires $O(n \log n)$ operations using quicksort. Thanks to the idea of incremental computation introduced in the IMHM [14], the procedure of graph construction for V can be implemented in $O(kn)$ since the length of an arc can be computed based on the previous arc length in $O(1)$, excepting the length of the first arc $L_{(0,k)}$ which is computed in $O(k)$ steps. Phase 3 is also accomplished in $O(n + kn)$ since the graph is a Directed Acyclic Graph (DAG) [21]. The last phase of the IMHMI has to visit each record once and is completed in $O(n)$ operations. Moreover, the space complexity of the IMHMI is in $O(n)$ that is used for storing the *parent* and *cost* vectors, each with $n + 1$ elements.

Notably, all computations of the IMHMI are performed in the integer space so the proposed method does not lose its stability as in the IMHM. In fact, for very large datasets of multi-million observations, SSE of IMHM may be different from SSE of MHM since numerical round-off errors are propagated during incremental real-value computations. A trivial approach to decrease the effect of the problem may be to periodically switch off the incremental computation of arc lengths in IMHM. However, the detailed solution and its impact on the quality of the protected dataset are out of the scope of this paper and are left for future research.

Regarding the trick that is used to ignore some arcs and terminate the inner loop (Line 21), generally, the probability of execution of the **break** increases for larger values of k, since more nodes are under the underlying arc from v_s to v_e that may contribute in the shortest path.

Noteworthy, multiplications by 2 in Equations (3) and (4) can be implemented using arithmetic left-shift operators in a low-level language to speed-up the process. Similarly, integer divisions by 2 in Equation (2) can be computed efficiently using arithmetic right-shift operators. Additionally, the frequent value of $cost_s$ can be stored in a simple variable before entering the inner loop, to reduce the execution time, since it

[3] Both items can be simply proved using expansion of the $\sum$ operator and algebraic substitutions.

[4] It is notable that the trick can also be used to accelerate the IMHM.

is not changed within the loop. However, these tricks are not shown in Algorithm 1 for brevity and clarity.

4 Experimental Results

In this section, the evaluation results of the IMHMI are presented. The experiments are conducted on a machine with a G3220 3.00 GHz CPU, 16 GB of RAM, running Windows 10 operating system. The algorithm is implemented in C++ and compiled with Microsoft Visual C++ compiler version 19.16.27030.1. In Section 4.1, the proposed algorithm is evaluated on multiple synthetic univariate datasets, while the results on some real-world multivariate datasets are reported in Section 4.2.

4.1 Application of the IMHMI for Univariate Microaggregation

In this section, the IMHMI is assessed on synthetic datasets of n random integer values with uniform distributions in $[-n/2, n/2]$ ($500\,000 \leq n \leq 20\,000\,000$). For each dataset size, the experiment is repeated 30 times with different initial seeds to produce various datasets. Table 1 shows the results of MHM, IMHM, and the IMHMI for different group sizes [5]. In almost all experiments, the IMHMI provides an improved SSE in compare with the SSE of the naïve application of the assignment of IMHM. For instance, for $n = 20M$ and $k = 4$, the average of relative improvements is about 0.12%. This small value is interesting regarding the optimal property of MHM. Additionally, the results confirm that IMHM and IMHMI are considerably faster than MHM. The IMHM is usually less time-consuming than the IMHMI, probably due to simpler algebraic computations of arc lengths in IMHM. However, the differences are usually negligible even for large datasets. Totally, the IMHMI performs better than the general univariate microaggregation algorithm MHM in terms of both information loss and runtime in all cases.

4.2 Application of the IMHMI for Multivariate Microaggregation

In this section, the IMHMI is applied to some real-world multivariate datasets. These datasets are cited in similar studies such as [16, 18, 22] and introduced in Table 2.

The IMHMI assumes that the observations are integer values that can be ordered. Additionally, protected attributes (i.e., the computed centroids by the IMHMI) has to be integer quantities. Therefore, a preprocessing step is required before applying the IMHMI on datasets of Table 2: (1) standardizing the attributes [6], (2) applying the ordering heuristic, and (3) scaling and rounding the values to integer quantities. As mentioned in Section 2, the sorting of multidimensional datasets is not well-defined for microaggregation problem. Therefore, some heuristic has to be used to order all records. The univariate algorithm then is applied to the ordered dataset. In the experiments of this section, 4 different heuristics are used to order multivariate records, i.e., the nearest point next (NPN) [16], the projection method based on PCA [17], the TSP ordering introduced in the FDM [19], and the projection method based on Z-score [17]. The first record in NPN and TSP heuristics is the farthest one from the centroid of the whole dataset. Additionally, all values are multiplied by 100 and rounded to the nearest integer to produce integer observations, ready to be used by the IMHMI. Table 3 shows SSE of IMHM and IMHMI for different group sizes k and ordering heuristics. It is notable that the error incurred by a multivariate microaggregation algorithm is the sum of errors of all attributes. This is shown in matrix form in Equation (5), where $x_{ij}, G_j \in \mathbb{Z}^d$ and d is the dimension of the dataset. Accordingly, implementations of IMHM and IMHMI are changed slightly to sum the SSE of each attribute and return it as the final computed SSE.

$$SSE = \sum_{j=1}^{g} \sum_{i=1}^{n_j} (x_{ij} - G_j)^T (x_{ij} - G_j) \qquad (5)$$

The results in Table 3 demonstrate the superiority of the IMHMI for different privacy parameters k and ordering heuristics. For instance, in $k = 3$ and using the NPN ordering, IMHM results in $SSE = 2\,251\,231$ for EIA, while the IMHMI improves it to $SSE = 2\,251\,217$. In all experiments, SSE increases when k is incremented. Additionally, in almost all experiments, the TSP heuristic produces the most useful protected dataset in terms of SSE. This fact is addressed in previous studies [19, 23].

Figures 1 and 2 illustrate the runtime of different methods for vlCensus and forest datasets, respectively. The privacy parameter k ranges from 2 to 50. The figures show that IMHM and IMHMI are much faster than MHM, especially for larger values of k. This confirms the methods are more interesting for practical usages and large datasets.

[5] Please recall that IMHM is an improved implementation of MHM. Therefore, its output is the same as MHM in terms of SSE and only the error of the output of IMHM is reported in Table 1.

[6] Standardization transforms each attribute to have a mean of zero and a standard deviation of one.

Table 1. The average and standard deviation of SSE and runtime of the (I)MHM and IMHMI for different group sizes and synthetic datasets

k	$n(\times 10^6)$	SSE		runtime (sec)		
		IMHM	IMHMI	MHM	IMHM	IMHMI
2	0.5	162 179.73 ± 238.56	162 047.27 ± 236.22	0.04 ± 0.01	0.04 ± 0.01	0.04 ± 0.00
	2	648 522.30 ± 688.32	648 000.27 ± 694.47	0.16 ± 0.01	0.15 ± 0.02	0.18 ± 0.02
	5	1 620 938.80 ± 1115.41	1 619 643.20 ± 1114.77	0.45 ± 0.07	0.37 ± 0.02	0.43 ± 0.02
	10	3 241 856.10 ± 1384.97	3 239 297.70 ± 1370.06	0.89 ± 0.14	0.73 ± 0.03	0.84 ± 0.03
	20	6 484 599.77 ± 1715.31	6 479 481.30 ± 1704.58	2.00 ± 0.29	1.47 ± 0.07	1.60 ± 0.22
3	0.5	358 295.97 ± 575.57	357 851.10 ± 568.23	0.03 ± 0.01	0.03 ± 0.01	0.03 ± 0.01
	2	1 433 331.63 ± 1306.13	1 431 515.77 ± 1290.68	0.13 ± 0.02	0.10 ± 0.01	0.14 ± 0.02
	5	3 583 541.17 ± 1995.62	3 579 080.50 ± 2007.78	0.37 ± 0.08	0.25 ± 0.02	0.33 ± 0.01
	10	7 166 859.33 ± 2017.97	7 157 999.27 ± 2022.67	0.74 ± 0.14	0.50 ± 0.05	0.65 ± 0.03
	20	14 334 838.73 ± 3225.10	14 317 135.07 ± 3246.19	1.28 ± 0.16	1.02 ± 0.10	1.29 ± 0.24
4	0.5	653 519.13 ± 969.39	652 297.20 ± 967.96	0.04 ± 0.01	0.03 ± 0.01	0.04 ± 0.01
	2	2 613 360.43 ± 2242.07	2 608 464.90 ± 2244.21	0.17 ± 0.04	0.11 ± 0.01	0.15 ± 0.01
	5	6 533 393.23 ± 3173.83	6 521 118.30 ± 3122.51	0.44 ± 0.06	0.28 ± 0.02	0.38 ± 0.02
	10	13 067 459.40 ± 4237.35	13 042 940.17 ± 4206.25	0.79 ± 0.10	0.53 ± 0.01	0.76 ± 0.03
	20	26 136 766.10 ± 4730.61	26 087 733.07 ± 4741.18	1.52 ± 0.15	1.15 ± 0.16	1.51 ± 0.23
5	0.5	1 026 032.43 ± 1320.03	1 025 130.97 ± 1308.71	0.05 ± 0.01	0.03 ± 0.00	0.04 ± 0.01
	2	4 102 874.43 ± 2353.69	4 099 298.37 ± 2343.45	0.24 ± 0.06	0.12 ± 0.01	0.18 ± 0.02
	5	10 255 386.17 ± 4593.16	10 246 432.03 ± 4590.06	0.48 ± 0.04	0.30 ± 0.02	0.43 ± 0.02
	10	20 513 379.17 ± 5343.78	20 495 482.20 ± 5292.48	0.97 ± 0.06	0.58 ± 0.02	0.88 ± 0.05
	20	41 025 516.20 ± 7020.95	40 989 651.23 ± 6945.41	1.95 ± 0.20	1.19 ± 0.11	1.70 ± 0.08
10	0.5	4 176 390.33 ± 4697.34	4 175 597.67 ± 4692.73	0.13 ± 0.02	0.05 ± 0.01	0.07 ± 0.01
	2	16 706 812.27 ± 9073.71	16 703 655.80 ± 9045.98	0.47 ± 0.05	0.18 ± 0.02	0.27 ± 0.02
	5	41 761 972.20 ± 13 314.56	41 754 107.50 ± 13 320.37	1.16 ± 0.09	0.44 ± 0.02	0.66 ± 0.04
	10	83 524 960.37 ± 25 653.67	83 509 196.83 ± 25 677.09	2.24 ± 0.09	0.86 ± 0.02	1.32 ± 0.06
	20	167 063 242.90 ± 22 794.65	167 031 668.43 ± 22 830.84	4.53 ± 0.26	1.80 ± 0.16	2.62 ± 0.10

Table 2. Numerical datasets to evaluate the IMHM.

dataset	records (n)	attributes (d)
Tarragona	834	13
census	1 080	13
EIA	4 092	11
vlCensus	149 642	13
forest	581 012	10

Table 3. *SSE* of the IMHM and IMHMI for different group sizes and real-world datasets sorted by various ordering heuristics

k	dataset	NPN		PCA		TSP		Z-score	
		IMHM	IMHMI	IMHM	IMHMI	IMHM	IMHMI	IMHM	IMHMI
2	Tarragona	13 260 893	13 260 893	17 488 857	17 488 857	10 147 342	10 147 342	22 644 761	22 644 761
	census	4 898 186	4 898 186	25 127 104	25 127 104	4 297 367	4 297 367	24 239 423	24 239 423
	EIA	1 131 223	1 131 187	48 503 537	48 503 537	1 112 755	1 112 718	46 623 390	46 623 386
	vlCensus	117 070 807	117 070 399	3 037 824 335	3 037 824 335	102 620 354	102 620 018	3 147 494 619	3 147 494 614
	forest	151 880 165	151 879 920	16 154 540 797	16 154 540 794	134 999 818	134 999 608	19 821 731 976	19 821 731 974
3	Tarragona	18 979 346	18 979 346	24 908 172	24 908 172	17 767 942	17 767 942	29 258 952	29 258 952
	census	8 711 694	8 711 694	34 230 297	34 230 297	7 417 415	7 417 415	33 934 143	33 934 143
	EIA	2 251 231	2 251 217	69 470 119	69 470 119	2 142 632	2 142 621	66 341 938	66 341 938
	vlCensus	219 553 127	219 552 881	4 185 892 936	4 185 892 934	179 673 467	179 673 284	4 341 859 572	4 341 859 570
	forest	285 566 482	285 566 364	22 208 868 055	22 208 868 055	247 296 944	247 296 813	27 244 425 745	27 244 425 741
4	Tarragona	23 492 012	23 492 005	27 329 082	27 329 081	23 777 988	23 777 988	33 481 594	33 481 594
	census	12 568 049	12 568 049	39 531 564	39 531 564	10 054 660	10 054 660	38 419 143	38 419 143
	EIA	3 097 443	3 097 435	81 844 938	81 844 938	3 366 553	3 366 535	75 242 168	75 242 168
	vlCensus	300 158 181	300 157 877	4 783 707 343	4 783 707 339	245 655 559	245 655 209	5 010 652 023	5 010 652 016
	forest	412 900 753	412 900 582	25 412 896 190	25 412 896 188	352 477 117	352 476 943	31 186 463 064	31 186 463 062
5	Tarragona	30 502 975	30 502 975	32 186 331	32 186 331	26 913 057	26 913 057	34 714 172	34 714 172
	census	15 503 503	15 503 503	42 483 280	42 483 280	12 179 645	12 179 645	41 403 339	41 403 339
	EIA	4 499 990	4 499 969	90 920 522	90 920 522	4 806 200	4 806 187	81 795 250	81 795 243
	vlCensus	387 086 392	387 086 119	5 155 198 132	5 155 198 132	307 706 315	307 706 076	5 398 854 742	5 398 854 710
	forest	538 556 934	538 556 800	27 421 550 342	27 421 550 321	453 679 939	453 679 789	33 665 468 056	33 665 468 050
10	Tarragona	41 925 425	41 925 425	40 163 339	40 163 339	40 565 215	40 565 215	40 857 047	40 857 045
	census	28 375 732	28 375 732	48 900 658	48 900 658	22 312 807	22 312 807	49 213 517	49 213 517
	EIA	11 225 343	11 225 342	107 512 160	107 512 160	12 221 589	12 221 582	100 077 948	100 077 946
	vlCensus	706 520 526	706 520 282	6 002 226 115	6 002 226 100	567 127 411	567 127 133	6 270 017 277	6 270 017 247
	forest	1 129 068 467	1 129 068 243	31 695 685 963	31 695 685 955	927 517 404	927 517 192	38 932 239 427	38 932 239 424

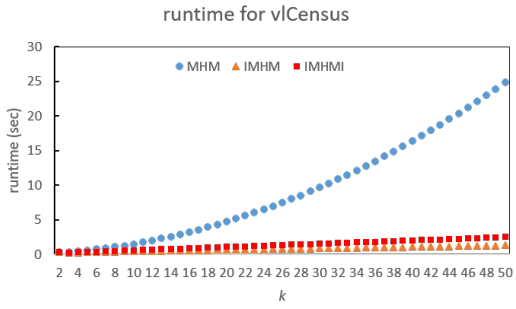

Figure 1. Runtime of different microaggregation methods for vlCensus dataset.

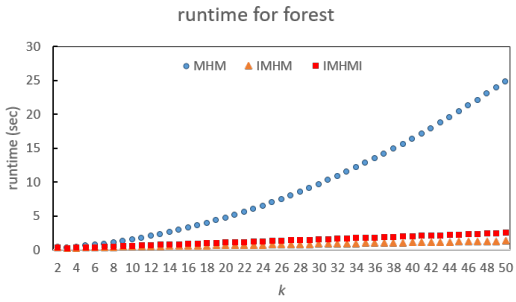

Figure 2. Runtime of different microaggregation methods for forest dataset.

5 Conclusions

This paper shows that the naïve application of the optimal univariate microaggregation MHM does not necessarily produce the best partition for integer datasets. An adapted algorithm IMHMI which benefits from incremental computations is proposed and implemented. The results confirm the superiority of the IMHMI in compare with similar methods, the MHM and IMHM in terms of data quality. While IMHMI focuses on pure integer datasets, it can also be used for other cases such as anonymizing graph degree sequences to evaluate how IMHMI improvements can affect the utility measures of anonymized graphs. More studies in this direction are needed.

Acknowledgment

The author would like to thank the editor and anonymous reviewers for all of their constructive and insightful comments about this study.

References

[1] Leon CRJ Willenborg and Ton De Waal. *Elements of statistical disclosure control*, volume 155. Springer Verlag, 2001.

[2] L. Sweeney. *k*-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[3] J. Domingo-Ferrer, A. Solanas, and A. Martinez-Balleste. Privacy in statistical databases: *k*-anonymity through microaggregation. In *Proceedings of International Conference on Granular Computing*, pages 774–777. IEEE, 2006.

[4] William E Winkler. Re-identification methods for masked microdata. In *Privacy in statistical databases*, pages 216–230. Springer, 2004.

[5] Josep Domingo-Ferrer and Vicenç Torra. Ordinal, continuous and heterogeneous *k*-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212, 2005.

[6] Reza Mortazavi and Saeed Jalili. Enhancing aggregation phase of microaggregation methods for interval disclosure risk minimization. *Data Mining and Knowledge Discovery*, 30(3):605–639, 2016.

[7] François Rousseau, Jordi Casas-Roma, and Michalis Vazirgiannis. Community-preserving anonymization of graphs. *Knowledge and Information Systems*, pages 1–29, 2017.

[8] Reza Mortazavi and Seyyedeh Hamideh Erfani. An effective method for utility preserving social network graph anonymization based on mathematical modeling. *International Journal of Engineering*, 31(10):1624–1632, 2018.

[9] Jordi Soria-Comas and Josep Domingo-Ferrer. Differentially private data publishing via optimal univariate microaggregation and record perturbation. *Knowledge-Based Systems*, 153:78 – 90, 2018.

[10] A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Commission for Europe*, 18(4):345–354, 2001.

[11] S.L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044, 2003.

[12] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 93–106. ACM, 2008.

[13] Jordi Casas-Roma, Jordi Herrera-Joancomartí, and Vicenç Torra. k-Degree anonymity and edge selection: improving data utility in large networks. *Knowledge and Information Systems*, 50(2):447–474, 2017.

[14] Reza Mortazavi, Saeed Jalili, and Hojjat Gohargazi. Multivariate microaggregation by iterative optimization. *Applied Intelligence*, 39(3):529–544, 2013.

[15] J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowl-*

edge and Data Engineering, 14(1):189–201, 2002.

[16] Josep Domingo-Ferrer, Antoni Martínez-Ballesté, Josep Maria Mateo-Sanz, and Francesc Sebé. Efficient multivariate data-oriented microaggregation. *The VLDB Journal*, 15(4):355–369, 2006.

[17] Jordi Nin, Javier Herranz, and Vicenç Torra. On the disclosure risk of multivariate microaggregation. *Data & Knowledge Engineering*, 67(3):399 – 412, 2008.

[18] David Rebollo Monedero, Ahmad Mohamad Mezher, Xavier Casanova Colomé, Jordi Forné, and Miguel Soriano. Efficient k-anonymous microaggregation of multivariate numerical data via principal component analysis. *Information Sciences*, 503:417 – 443, 2019.

[19] Reza Mortazavi and Saeed Jalili. Fast data-oriented microaggregation algorithm for large numerical datasets. *Knowledge-Based Systems*, 67:195 – 205, 2014.

[20] Cynthia Dwork. *Differential Privacy*, pages 338–340. Springer US, Boston, MA, 2011.

[21] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[22] Reza Mortazavi and Saeed Jalili. A novel local search method for microaggregation. *ISeCure*, 7(1):15 – 26, 2015.

[23] B. Heaton. *New Record Ordering Heuristics for Multivariate Microaggregation*. PhD thesis, Nova Southeastern University, 2012.

Reza Mortazavi received the Ph.D. degree in computer engineering from Tarbiat Modares University (TMU) in 2015. He received the M.Sc. degree in software engineering from the Khaje Nasir Toosi University of Technology in 2007, and the B.Sc. degree in software engineering from Shahid Beheshti University in 2005. His main research interests are Privacy-Preserving Data Publishing (PPDP) and Statistical Disclosure Control (SDC) methods.