Appl Intell (2013) 39:529–544
DOI 10.1007/s10489-013-0431-y

Multivariate microaggregation by iterative optimization

Reza Mortazavi · Saeed Jalili · Hojjat Gohargazi

Published online: 12 April 2013

Abstract Microaggregation is a well-known perturbative approach to publish personal or financial records while preserving the privacy of data subjects. Microaggregation is also a mechanism to realize the k-anonymity model for Privacy Preserving Data Publishing (PPDP). Microaggregation consists of two successive phases: partitioning the underlying records into small clusters with at least k records and aggregating the clustered records by a special kind of cluster statistic as a replacement. Optimal multivariate microaggregation has been shown to be NP-hard. Several heuristic approaches have been proposed in the literature. This paper presents an iterative optimization method based on the optimal solution of the microaggregation problem (IMHM). The method builds the groups based on constrained clustering and linear programming relaxation and fine-tunes the results within an integrated iterative approach. Experimental results on both synthetic and real-world data sets show that IMHM introduces less information loss for a given privacy parameter, and can be adopted for different real world applications.

Keywords Microaggregation · Microdata protection · k-Anonymity · Constrained clustering · Iterative optimization

R. Mortazavi · S. Jalili (✉) · H. Gohargazi
Electrical and Computer Engineering Faculty, Tarbiat Modares University, Tehran, Iran
e-mail: sjalili@modares.ac.ir

R. Mortazavi
e-mail: r.mortazavi@modares.ac.ir

H. Gohargazi
e-mail: h.gohargazi@modares.ac.ir

1 Introduction

Continuous advances in computer technologies enable corporations to collect an enormous amount of personal data, which are valuable resources for researchers, the public, and governmental agencies. However, before dissemination of such data, some procedures must be applied to make the data anonymous. Multiple models and mechanisms have been proposed for the requirements of this anonymity in the database [1–3], data mining [4, 5], and statistics literature [6–8]. The protection is achieved at the expense of the reduction of the utility of protected data for the eligible users such as researchers, enforcing them to use special methods to extract knowledge from published data [9]. Different approaches to implement a privacy model must trade-off between what is needed for the privacy level specified by the model and the utility of protected data produced by the anonymizing procedures.

A basic computational privacy model is k-anonymity [1], in which the record of a data subject must be the same as at least $k-1$ other records in the data set. Several sophistications of k-anonymity and its variants have been proposed in the database security literature such as (α, k)-anonymity [10], (L, α)-diversity [11], and (p, α)-sensitive k-anonymity [12]. Protecting statistical databases is also worth considering for statistical agencies and is studied as Statistical Disclosure Control (SDC) in the statistics literature. Microaggregation is a family of perturbative protection methods for microdata publishing introduced in SDC [13, 14], with applications in computer science such as Privacy Preserving Data Publishing (PPDP) [6, 15] and Location-Based Services (LBS) [16]. In this approach, records are first partitioned into clusters of size at least k, and then a good measure of central tendency for each cluster such as the mean, median, or mode [17] of the cluster is

released. The result is called protected data set and offers no information by itself to distinguish any data subject from a set of at least $k-1$ other data subjects. Microaggregation is essentially limited to numeric attributes to calculate the mean of a cluster (leading to its centroid), but there are also categorical extensions [18].

Information Loss (*IL*) measures the utility of published protected data after microaggregation. Lower *IL* means less distortion of the original records and thus implies more utility. To reduce *IL*, the microaggregation procedure must group *similar* and *sufficient* records. Similar records form a more compact cluster so the calculated centroid is a more appropriate representative. Additionally, there must be an accurate number of records within a cluster between k and $2k-1$ to satisfy the k-anonymity requirement and preserve the most utility, respectively.

This paper presents a novel approach for multivariate microaggregation based on an iterative optimization method. A solution based on relaxed integer programming with the privacy and utility requirements as constraints is proposed. In addition, the MHM algorithm [19] is used during the optimization to determine the number of clusters. Experimental results on the standard SDC data sets show the effectiveness of the algorithm in comparison with the state of the art approach, MDAV [17]. For the same level of privacy defined by the parameter k of the k-anonymity model, the proposed method reduces the *IL* measure for synthetic data sets by up to 20 % in comparison with MDAV.

It is notable to say that while aggregation may prevent re-identification, it doesn't necessarily protect against information disclosure. This is due to the attacker model in k-anonymity in which, he/she is interested to know the exact record of a data subject within the published data. Although there are some extensions of microaggregation to more complicated models [20–22], we have used its main functionality to realize k-anonymity. In fact, the privacy model by its own does not necessarily provide a guarantee; and it is common practice in SDC to review the protected data set by an expert after anonymization [23]. There are some probabilistic measures which are used to assess the disclosure risk of a protected data set regarding the original data set [24, 25].

The rest of this paper is organized in the following manner. Section 2 is intended to define the problem, while Sect. 3 offers a brief survey of previous works on the microaggregation methods. Section 4 points out the proposed iterative optimization. Section 5 is devoted to the experimental results, and Sect. 6 concludes the paper.

2 Problem definition

Microaggregation is the most recent perturbative approach to mask microdata for protecting data subjects against reidentification in secure data publishing [24, 26–28]. It is usually modeled as a clustering mechanism with cluster size constraints. The microaggregation problem may be formulated as follows [14, 19, 29]. A data set T with n records and d numeric attributes is specified, where each record can be considered as a vector in a d-dimensional space. Given a privacy parameter k, a microaggregation method, partitions T into c clusters, each of which has at least k records, as cluster members, and then replaces the cluster records with the centroid of the cluster.

Satisfying the privacy requirement involves distortion of data and a decrease in the utility which is measured by information loss (*IL*). Formally speaking, let $n_i \geq k$ denotes the number of records in the ith cluster, C_i. The centroid of the cluster denoted by $\overline{x}_i$, is calculated as the mean of the cluster members. The centroid of the whole data set T is the average of all involved records, $\overline{x}$. This paper adopts the most common definition for information loss [30], $IL = SSE/SST \times 100\ \%$, where *SSE* is the sum of within-cluster squared errors and *SST* is the sum of squared error for the whole data set T, as in Eq. (1).

$$\begin{aligned} SSE &= \sum_{i=1}^{c}\sum_{j=1}^{n_i}(x_{ij}-\overline{x}_i)^T(x_{ij}-\overline{x}_i) \\ SST &= \sum_{i=1}^{c}\sum_{j=1}^{n_i}(x_{ij}-\overline{x})^T(x_{ij}-\overline{x}) \end{aligned} \tag{1}$$

where x_{ij} is the jth member of the ith cluster. Since the *SST* value is only dependent on the given data set T, the microaggregation problem can be formulated as follows. Given a data set T, partition it into clusters $C = \{C_1, \ldots, C_c\}$ in a way that the *SSE* value is minimized such that the constraint $|C_i| \geq k$ for any $C_i \in C$ is met. Unfortunately, there is no optimal polynomial solution for the problem unless $d = 1$ [19]. It has been proven that the problem is NP-hard for a general multivariate data set [31].

Generally, when the number of clusters decreases, more records appear in the clusters and the within-cluster error increases which results in more distortion and a larger value for *IL*. However, there may be cases in which lower values of *IL* may be produced by fewer clusters. Section 4.3 shows a more illustrative description of the situation.

3 Microaggregation methods

Table 1 shows some of the most known microaggregation methods along with their classifications. Most proposed algorithms in the literature have some tuning parameters, where the data publisher must somehow set them for the algorithm to operate efficiently. These parameters are also shown in Table 1. Microaggregation methods can be classified into univariate and multivariate based on the data set

Table 1 Classification of microaggregation algorithms

Method	Parameters/Options set by Publisher	Classification*	Ref.
Particular variable	Index of the selected variable	U/F	[14]
Principal component	–	U/F	
Sum of z-score	–	U/F	
MHM	–	U/D	[19]
MHM-Based (NPN, MD, ...)	Multidimensional Sorting Criteria	M/D	[34]
MHM-Based (TSP)	Tour construction heuristics, first record for conversion from tour to path	M/D	[35]
MDAV	–	M/F	[17]
CBFS	–	M/F	[36]
V-MDAV	λ: gain factor	M/D	[37]
TFRP	Maximum size of a group in phase I	M/D	[38]
MST-Based	Method to partition oversized groups	M/D	[36]
Genetic Algorithm	Pre-partitioning size, GA parameters such as Crossover and Mutation rate	M/D	[39]
DBA-2P	p: number of lowest density groups formed initially	M/D	[40]
PCL	Initial probability constraints	M/D	[41]
GSMS-T2	Maximum size of a group in phase I	M/D	[42]

* U: univariate, M: multivariate, F: fixed size, D: data oriented

Fig. 1 MDAV microaggregation algorithm

1. **INPUT**: T(data set), k (privacy parameter)
2. **OUTPUT**: A (Assignment of records to clusters)
3. **FUNCTION** [A] = MDAV (T: data set, k: integer)
4. Compute the centroid $\bar{x}$ of the data set.
5. Find the most distant record r from $\bar{x}$. Form a cluster containing r and its $k-1$ neighbors. Set aside this cluster from the data set.
6. Find the most distant record s from r. Form another cluster containing s and its $k-1$ neighbors. Set aside this cluster from the data set.
7. If there are at least $2k$ records remaining (unassigned), repeat steps 4 to 6.
8. If there are between k and $2k-1$ records unassigned, form a new cluster containing all of them and return the assignment of records to clusters.
9. If there exist at most $k-1$ unassigned records, assign each of them to the closest cluster centroid.
10. Return the assignment of records to clusters.
11. **END FUNCTION**

dimension. The case of univariate microaggregation can be solved optimally in polynomial time based on the shortest path problem [19], called MHM method, however the multivariate variant is shown to be NP-hard [31]. There are some approaches that use the univariate solution to solve the multivariate problems such as using the first principal component, sum of z-scores [24], or even considering only a particular attribute [32, 33]. In addition, there are some multivariate extensions of MHM, where records are ordered based on various heuristics such as Nearest Point Next (MHM-NPN) [34] and next point in a TSP tour [35].

Moreover, Microaggregation methods can be further categorized into fixed size and data oriented [30]. In fixed size methods, all clusters have the same size k, but one cluster with a size between k and $2k-1$ may exist. On the other hand, data oriented methods result in variable size clusters with at least k records. The cluster size is dependent on the distribution of records, which are near the cluster center in order to preserve natural data aggregates. Generally, data oriented methods result in higher protection level and less information loss, but fixed size methods are more efficient [14, 43].

Domingo Ferrer and Torra proposed Maximum Distance to Average Vector (MDAV) [17], a well-known fixed size microaggregation method. MDAV is implemented as a technique in the μ-ARGUS software and is the most widely-used microaggregation method [44]. The pseudo-code of MDAV is shown in Fig. 1. MDAV first finds the most distant record, say r, from the centroid of the data set and a new search for another most distant record from r, say s, is accomplished subsequently. The next step is to build two clusters including r and s and their $k-1$ nearest records respectively. These two clusters are microaggregated and removed from the data set. The latter steps are repeated until less than $2k$ records remain. The remaining records form a new cluster or are assigned to their respective nearest groups.

Another algorithm similar to MDAV, called Centroid-Based Fixed-Size (CBFS) microaggregation, is based on the centroid idea and introduced in [36]. CBFS acts like MDAV, but forms only one cluster in each iteration.[1] Variable-sized MDAV, called V-MDAV is also presented in [37]. V-MDAV tries to extend the cluster up to $2k-1$ records. The addition of a new record to a cluster is based on a user-defined threshold parameter γ, named gain factor. After each cluster formation, the nearest unassigned record to any cluster member is found, say e_{min}, and its distance is computed, d_{in}. Suppose d_{out} denotes the distance from e_{min} to its nearest unassigned record. The record e_{min} will be added to the cluster if the cluster size is lower than $2k$ after addition and $d_{in} < \gamma.d_{out}$. Experimental results showed that setting γ close to zero is effective for scattered data sets, however using a single γ value for the whole data set will not give good results if the data set is a mixture of scattered and clustered data [37]. Chang et al. reported Two Fixed Reference Points (TFRP) method [38]. Let G_{min} and G_{max} denote the minimum and maximum vectors over all attributes of records in the data set, respectively. The first step of TFRP is to calculate these extremes as reference points, say R_1 and R_2. The second step is cluster formation. A cluster consisting of k records is formed around r, which is the most distant record from R_1. Another cluster of size k is formed with respect to the most distant record from R_2, say s, and its $k-1$ nearest neighbors. The last two steps are repeated iteratively until fewer than k records remain, which are added to their nearest clusters. A post processing step may be conducted to refine the clusters. Notably, fixing R_1 and R_2 as reference points makes TFRP an efficient approach and comparable to the most recent works in the field [42].

Clustering records based on a tree representation is discussed in [36], where the microaggregation method is based on the computation of the Minimum Spanning Tree (MST). First a minimum spanning tree is constructed based on records in the data set where records are vertices and the distances between records as their corresponding edge weights. The longest edge will be removed from this MST recursively, if all the remaining tree clusters contain at least k records. This approach works effectively on well-separated clusters [26, 40].

When the data set is small ($n \leq 50$), an evolutionary algorithm may be conducted to explore the search space and produce a (possibly local) optimal result. Solanas et al. presented the assignment of records to clusters by a chromosome containing n genes where the value of the ith gene is the cluster number to which the ith record is assigned to [39]. During evolution of the population, special care must be taken to avoid generating a chromosome which violates the privacy or utility requirements in terms of the minimum or maximum genes with the same cluster number, respectively. The authors recommended using a fixed-size microaggregation beforehand with $k = 50$, and then apply GA on each of the resulting clusters. DBA [40] is an approach based on a density heuristic. The authors started aggregation from more dense regions of the data set and then performed a refinement phase to check whether to decompose a cluster or leave it intact.[2] The authors have also proposed an extended version of their work named DBA-2P, including a preprocessing step. In this algorithm, p clusters are formed from those records with the lowest densities and their respective $k-1$ nearest neighbors. More recently, a microaggregation method inspired by the design of distortion-optimized quantizers, called Probability-Constrained Lloyd (PCL) is published [41]. The authors used the idea of k-means clustering algorithm,[3] with an additional probabilistic cost function term in the nearest-neighbor step assigned to a cluster.[4] In this scheme, increasing the cost of a given cluster, without any change in the centroids and the cost of the other clusters, will reduce the number of records assigned to it. The authors reported a reduction in IL about 15 % to 27 % compared with MDAV for Gaussian and uniform distributions, respectively. Panagiotakis et al. proposed successive Group Selection algorithm based on sequential Minimization of SSE (GSMS) [42]. GSMS iteratively discards the cluster that minimizes the current SSE. To improve the method, the authors introduced GSMS-T2, in which a refining phase like the second phase of TFRP is used. GSMS-T2 produces comparable information loss regarding the other algorithms with similar time complexity. For a more thorough survey about microaggregation methods, interested readers are recommended to refer to [29, 45].

4 Iterative MHM-based microaggregation

This section presents the proposed method, iterative optimization based on optimal univariate microaggregation (IMHM). This study is restricted only to continuous attributes, however, there may be extensions to other data types [17]. Additionally, the main criterion in IMHM is to reduce information loss, so it is recommended for the most general case of offline anonymization in a reasonable time.

[1] The pseudo-code of CBFS is very similar to MDAV in Fig. 1, but line 6 is removed and the main loop is iterated until less than k unassigned records remain ($2k$ changes to k in line 7 of Fig. 1).

[2] The terms "merge" and "noMerge" are used in the DBA paper.

[3] Referred to as "Lloyd quantization design algorithm" in the original paper.

[4] Referred to as "cell" in the original paper.

4.1 Microaggregation as an optimization problem

The result of a microaggregation process is very similar to a conventional clustering algorithm in which all records are aggregated within some clusters. Nevertheless the difference is in the cluster size constraints. So our formalization approach in this section is like a k-means algorithm [46] adopted for microaggregation. After microaggregation, cluster centers are calculated based on the assignment of records to clusters, where centroids are a simple average of assigned records.

Given a set of clusters $C = \{C_1, C_2, \ldots, C_c\}$, the assignment of the jth record to the ith cluster (denoted by $b_{ij} = 1$) increases the information loss, which is denoted by w_{ij}. All records prefer to be assigned to the nearest clusters, so that information loss due to the aggregation is minimized. The IMHM uses $w_{ij} = \|C_i - X_j\|$ as the cost of an assignment, where C_i denotes the ith cluster center and X_j represents the jth record, both in d-dimensional space, and $\|D\| = \sqrt{D^T D}$. The microaggregation problem can be formulated as an optimization problem as Eq. (2).

$$
\begin{aligned}
F = \text{Minimize} & \sum_{i=1}^{c}\sum_{j=1}^{n} w_{ij} \cdot b_{ij} \\
\text{subject to} & \sum_{i=1}^{c} b_{ij} = 1 \qquad j \in \{1, \ldots, n\} \\
& k \le \sum_{j=1}^{n} b_{ij} < 2k \quad i \in \{1, \ldots, c\} \\
& b_{ij} \in \{0, 1\}.
\end{aligned}
\tag{2}
$$

The first constraint in Eq. (2) guarantees that all records are assigned to exactly one cluster, while the next inequality constraints are to satisfy the privacy and utility requirements of microaggregation. The variable of b_{ij} is an indicator which determines the assignment of the jth record to the ith cluster with the cost w_{ij}. A record is assigned to a cluster whenever its associated indicator variable is set, i.e., $b_{ij} = 1$.

Solving the optimization problem (2) in its original form requires exponential space and time [47] that restricts the solution's scalability. Fortunately, the constraint matrix has the property of being *totally unimodular* [48, 49]. A 0-1 matrix M is totally unimodular if the determinant of each square sub matrix of M is $-1, 0$, or 1. This property implies that the relaxed version of the problem ($b_{ij} \in [0, 1]$) has the same integral solution of the original integer programming [49]. The solution of such a relaxed version can be efficiently achieved via any generic optimization method such as the Simplex algorithm, where the solution may be found in polynomial time in practice [47].

Given the number of clusters c, the solution is guaranteed to satisfy the privacy conditions for all clusters. However, determining the exact number of cluster centroids in the domain space of the data set and even predicting the optimal number of clusters are hard (if even possible). So, the IMHM uses the result of the optimal univariate microaggregation iteratively—in the same manner of MDAV-MHM [34]—to determine the number of clusters, c.

4.2 Implementation of IMHM

The pseudo-code of IMHM is presented in Fig. 2. The inputs of the algorithm are the whole data set T containing $n = |T|$ vectors in d-dimensional space, and the privacy requirement k. All records with non-existent attributes are omitted. The outputs of the algorithm are the centroids and assignment of records to clusters. For example, assignment $[5] = 3$ means the fifth record (in the same order of the input data set T) is assigned to the third cluster.

First, a fixed size microaggregation algorithm is executed to initialize the centers and assignment. This step is required for *pre-partitioning* sizable data sets (see Sect. 5.2). Microaggregation is normally used by statistical agencies after "*blocking*" large data sets into smaller sub-data sets [34]. So, the IMHM pre-partitions large data sets into *NPB*[5] blocks of a predefined maximum size of records, *MAX_SIZE*. The IMHM uses the pre-partitioning based on the result of MDAV [17, 40] with the same k as the privacy requirement (line 4).[6] In line 5, all centroids are assigned a number using the Sort_Centers function (lines 30 to 35). In this function, the center of centroids is calculated (line 31) and the most distant centroid F, is selected (line 32). Then, all clusters are numbered[7] based on their respective Euclidean distances to F in an ascending manner (lines 33 and 34). The IMHM accepts one block of records in each execution, so if the underlying data set is large, it must be split into blocks, each of which contains some clusters of the MDAV algorithm (line 6). All cluster numbers in a block are successive and fall in a proper range to satisfy a maximum allowed number of records in a single execution of IMHM, *MAX_SIZE* (line 6). In this way, information loss due to this pre-partitioning is reduced. Without loss of generality, the remaining of this section assumes that there is only one block ($NPB = 1$), and the whole data set T satisfies $n = |T| \le MAX\_SIZE$.

The IMHM starts with the maximum number of clusters, i.e., $\lfloor n/k \rfloor$, and sets the *IL* of the current block to infinite (lines 9–11). Generally, more clusters results in less records within the clusters, and so the *SSE* and *IL* are decreased (see Sect. 4.3). In each iteration, the constraints and objective function of the program are constructed or updated

[5] Number of Pre-partitioned Blocks (*NPB*).

[6] We have also used a semi-random initialization, which is discussed in Sect. 5.4.

[7] Here, the index of each centroid is its implicit number.

Fig. 2 Pseudo-code of IMHM

```
1.  INPUT: T(data set), k (privacy parameter)
2.  OUTPUT: C (Centroids), A (Assignment)
3.  FUNCTION [C, A] = IMHM (T, k)
4.  [Initial_Centers, Initial_Assignment] = MDAV (T,k)
5.  Centroids = Sort_Centers (Initial_Centers)
6.  Blocks = Partition (T, Centroids, MAX_SIZE)
7.  Iteration = 0
8.  FOR EACH block B ∈ Blocks
9.     Current_Centroids = Get Centroids of B
10.    Current_Assignment = Get assignment of records in B  // initial assignment
11.    NClusters =|Current_Centroids|, Best_Block_IL = +∞
12.    REPEAT UNTIL a stopping criterion is satisfied // MAX_ITERATION or MIN_IL_CHANGE
13.       LPC = Form Constraints from NClusters, and k     // The LP Constraints
14.       LPO = Form Objective Function  // equation (2)
15.       Current_Assignment = Solve LP (LPO,LPC,{solution history}) // Call the optimization engine
16.       Calculate Current_Centroids and Current_Block_IL based on Current_Assignment
17.       IF (no improvement in Current_Block_IL)
18.          DO  // assignment refinement
19.             P = Construct_Path (B, DistRC, Current_Assignment)
20.             Use ImprovedMHM (P, k) and update Current_Centroids, DistRC and Current_Block_IL
21.          UNTIL a stopping criterion is satisfied    //  MAX_ITERATION or MIN_IL_CHANGE
22.          IF (no improvement in Current_Block_IL) BREAK the loop for the current block B
23.       END IF
24.       Update C and A, as the best centers and assignment found
25.       Iteration = Iteration + 1
26.    END REPEAT
27. END FOR
28. RETURN [C, A]
29. END FUNCTION
30. FUNCTION [Sorted_Centers] = Sort_Centers (Centroids)
31. CC = Calculate the center of Centroids
32. F = Find the most distant centroid from CC
33. D = Calculate distances of Centroids to F
34. Sorted_Centers = Centroids in the same order of Sort (D, "Ascending")
35. END FUNCTION
36. FUNCTION [Path] = Construct_Path (AllRecords, DistRC, Assignment)
37. IF one cluster  // trivial case
38.    Path = AllRecords; RETURN
39. CR = Calculate the center of AllRecords
40. F = Find the most distant record from CR
41. Current_Cluster = Assignment[F]
42. LOOP FOR EVER
43.    Current_Records = {X ∈ Current_Cluster}
44.    Next_Cluster = Find the nearest unassigned centroid to Current_Records
45.    Add X_i ∈ Current_Records to Path, in descending order of DistRC(X_i,Next_Cluster)
46.    IF more than one unassigned centroid is remained
47.       Current_Cluster = Next_Cluster
48.    ELSE
49.       Assign X_i ∈ remaining records, in ascending order of DistRC(X_i, Current_Cluster); RETURN
50.    END IF
51. END LOOP
52. END FUNCTION
```

(lines 13–14). For determining the assignment of records to clusters, IMHM performs like k-means [46]. Then the assignment, centroids, and IL of the current block are calculated and the consequent updates on C and A occur (lines 16, 24). Solving the program may use the solution of the last iteration, which is shown as the *solution history* (line 15). This makes a significant runtime improvement, because most records are assigned to the same clusters of the previous iteration. However, if the number of clusters is changed, optimization must be restarted. If the optimization stops further improvement (for example, trapped in a local optimum), a procedure similar to MDAV-MHM [34] is used iteratively to refine the assignment. To refine the assignment, a path P is constructed based on *Current_Assignment* (line 19). All records in the block B, distances of records to centroids *DistRC*, and the current assignment are passed to the Construct_Path function (see Sect. 4.4). In this function, the trivial case is handled in line 38 without any calculations. Lines 39–41 find the index of the most distant record F from the center of records, and sets its cluster index as

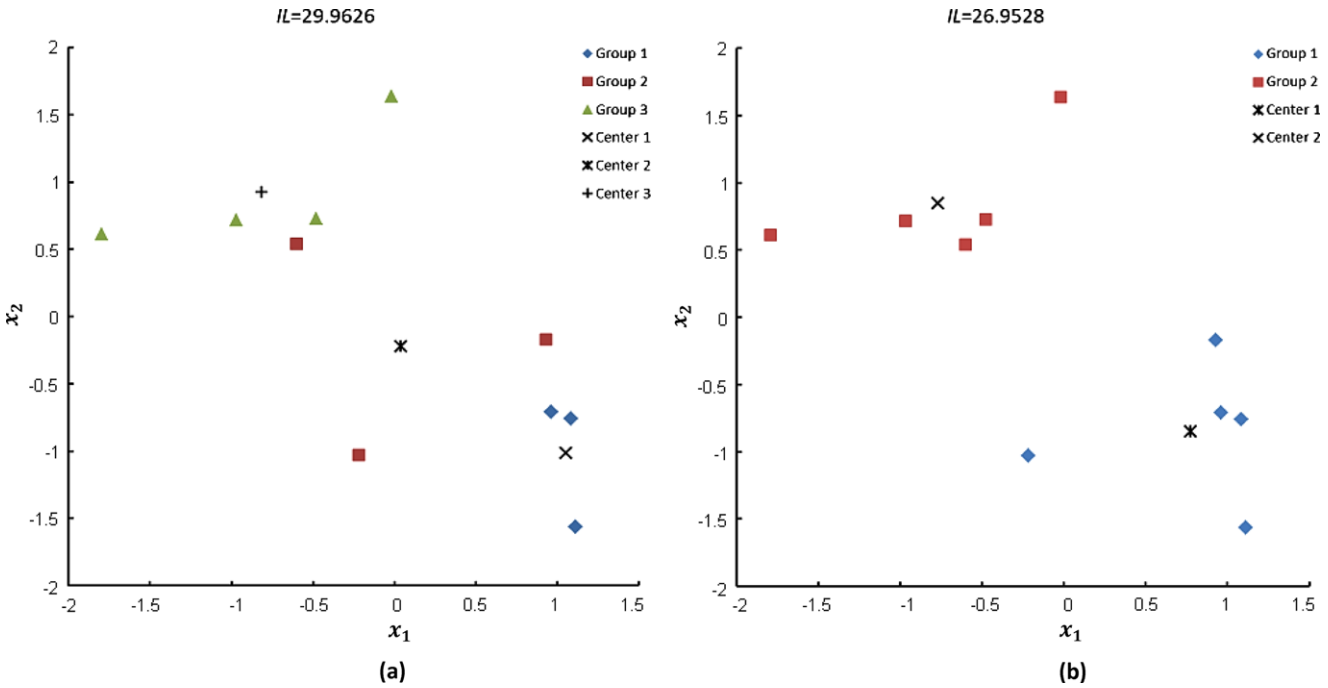


Fig. 3 Increasing the privacy and utility for $k = 3$, $IL = SSE/SST \times 100\ \%$ is used. *Filled shapes* represent 10 records in two-dimensional space (x_1, x_2), and *crossed lines* correspond for cluster centers

Current_Cluster. Then, the nearest centroid among all other unassigned[8] centroids to any records in *Current_Cluster* is selected as *Next_Cluster*. All records in *Current_Cluster* are added to *Path* based on their distances to *Next_Cluster*, while the nearest record is added at the last (line 45). If there exists more than one unassigned centroid, *Current_Cluster* is updated and the loop continues (line 47). All records in the last cluster are added to *Path* in an increasing order of their distances to the last centroid in the *Path* (line 49).

After the path construction, an improved version of MHM (see Sect. 4.4) is used to refine the assignment (line 20). In addition to reducing the *SSE* of the assignment, the procedure has two more important benefits: (1) it can get the IMHM out of local optimum, and (2) it can determine the number of clusters, adaptively. It is guaranteed to produce a clustering as well as the input. This makes the IMHM a data-oriented microaggregation method, which usually results in more utility for protected data set. The inner loop continues until a stopping criterion such as the minimum change in *IL*, is satisfied (line 21). If no improvement is achieved within the loop, the algorithm terminates the optimization for the current block in line 22.

There is a constraint for each cluster and a constraint for each record. Furthermore, there are $c \times n$ indicator variables, so the constraint matrix of IMHM has $(c+n)(c \times n)$ entries. However, there may be some heuristics to exclude a cluster and all of its members from participating in some iterations of the optimization process. This does not change the unimodularity property of the constraint matrix, and the same optimization methods can be used. However, using this performance tweak is left as a future work, especially for larger data sets.

[8] A centroid is called unassigned, if none of its members are assigned yet. Similarly, a remained record is a record not in the *Path*.

4.3 Determining the number of clusters, c

Given n records in T, and k as the privacy parameter, the size of a cluster must be between k and $2k-1$ to satisfy both the privacy and utility. Therefore, the number of clusters will be between $\lceil n/(2k-1) \rceil$ and $\lfloor n/k \rfloor$. Generally, more clusters implies less information loss because fewer records within each cluster are to be aggregated, and less distortion is incurred. However, there may be situations in which re-settling all cluster members and deleting the cluster reduces the information loss and at the same time offers more privacy. In other words, if the records in the data set are from two natural clusters of size between $3k/2$ and $2k-1$, they may be broken into three clusters satisfying the minimum size constraint and one containing records from each of the natural clusters that are far apart. For example, two possible clustering outputs for 10 records in a two-dimensional space (x_1, x_2) are drawn in Fig. 3. Part (a) of Fig. 3 depicts a cluster between two other clusters that can be broken up to decrease the *IL*, sketched in part (b) of Fig. 3.

The IMHM starts from the most preferable number of clusters, i.e., $\lfloor n/k \rfloor$ and determines the number of clusters

in an adaptive manner by sorting the records in the data set similar to MDAV-MHM and then applying an improved version of the MHM [34], which will be described in the next section.

4.4 Assignment refinement

As mentioned in Sect. 4.2, an assignment refinement phase is applied to determine c, and improve the assignment (lines 18–21 of Fig. 2). The refinement process loops over a path construction function followed by a procedure similar to the MHM algorithm [19]. In this section, we discuss the efficiency of these procedures.

First, the path construction function finds the most distant record which is done in $O(n)$.[9] The *Next_Cluster* can be found in $O(n)$. Sorting all records in the *Current_Cluster* requires $O(k \log k)$ computations. The loop iterates once for each cluster, so the total runtime of the function would be $(\max(n^2/k, n \log k))$. However, regarding the fact that there is no need to calculate the distances among centroids, and all of the distances between records and centroids are computed or updated before the path construction (for example in forming the objective function in line 14 of Fig. 2), the function can be executed more efficiently in comparison with other schemes introduced in [34].

The runtime of the MHM, $O(\max(n \log n, nk^2))$ for univariate microaggregation [19], may prohibit the usage of IMHM for larger values of k, which may be favorable for some applications (such as LBS [41]).[10] In this section, we introduce an enhancement to the runtime of MHM.

The idea is about to calculate the *SSE* of each group incrementally after the first group. The pseudo-code of the ImprovedMHM is presented in Fig. 4. In line 8, the graph $M(V, E)$ is initialized (for clarity, all vertices are denoted by the same index of their representative records from X_1 to $X_{|P|}$, along with a *dummy* record X_0 in the beginning). After the initialization, the *Centroid* and *SSE* of the first group are calculated (lines 12–13). The results are saved for later use (lines 14–15). Then, other records are added to the group until the group size reaches the limit of $2k-1$ or there are no more records to add. The weight of the next edge (i.e., from X_0 to X_{k+1}) after the weight of the first edge is calculated incrementally, and the respective centroid is also updated[11] (lines 23–26). For calculating the next weight and centroid for the group of records from X_1 to X_{k+1}, the values saved for the first group from X_0 to X_k are used (lines 17–19), and these values are updated (lines 20–21). After the construction of the graph, the shortest path from X_0 to $X_{|P|}$ is used to find the optimal assignment of records to clusters, regarding the provided path P (lines 31–38).

The ImprovedMHM algorithm runs[12] in $O(k + nk + n(k+1)) = O(nk)$. Only the first weight has to be calculated completely in $O(k)$, while all of the next weights and centroids may be calculated in $O(1)$. The shortest path can be calculated in $O(n + nk)$. So, the runtime of ImprovedMHM is $O(nk)$.

4.5 The convergence of IMHM

The IMHM consists of three main procedures: (i) the assignment of records to centers, (ii) updating the centroids, (iii) constructing a path and using ImprovedMHM. It is easy to verify that all these procedures do produce an assignment that is at least as good as their input, in terms of *IL* and the *SSE* is reduced iteratively. The optimality of the assignment of records to the fixed centers is due to the used objective function and optimality of the applied optimization method (i.e., the Simplex algorithm). The next procedure to update the centroids reduces *IL*, because the mean of a set of data points is the most suitable point to minimize *IL*. Formally, for a set of n data points X_i, the value of C which minimizes $\sum_{i=1}^{n} \|X_i - C\|^2$ is equal to $C = \sum_{i=1}^{n} X_i/n$, or the mean of the points. The iterative applications of the path construction and ImprovedMHM do not increase the *SSE*, because there is at least a path from X_0 to $X_{|P|}$ in $M(V, E)$, which can produce the same assignment as its input, thus the shortest path algorithm can find it (line 30 in Fig. 4). The IMHM converges to (local) optimum quickly, which is shown in the experiments of Sect. 5.

5 Experimental results

In this section, we evaluate the IMHM for different scenarios including tests on standard benchmark data sets along with three synthetic data sets in Sect. 5.1. Two large data sets are also evaluated in Sect. 5.2. The runtime and convergence speed of IMHM are discussed in Sect. 5.3. The final experiment in Sect. 5.4 is to show the feasibility of using a random initialization method against minimality attack [50].

A prototype of IMHM is implemented in MATLAB. All tests are conducted within MATLAB 2012a in Windows 7 32-bit operating system on a laptop PC with Intel Core i7 1.6 GHz CPU. The Simplex algorithm of MOSEK 6.0.0.126 [51] used as the optimization method. The code was not optimized for speed, except some tricks defined in the following. Thanks to the "Hot Start" feature of the engine applicable for the Simplex method, the result of the

[9] We assume, $NPB = 1$, i.e., $|B| = |T| = n$.

[10] Please note that another procedure reorders records in a path for ImprovedMHM algorithm (line 19 in Fig. 2), so the term $n \log n$ does not make sense here.

[11] Please see the Appendix.

[12] For simplicity, we removed the time complexity related to the dimension, d.

Fig. 4 The pseudo-code of ImprovedMHM

1. **INPUT**: order of all records in a path P, privacy parameter k
2. **OUTPUT**: Optimal assignment of records to clusters regarding the provided path A
3. **FUNCTION** [A] = ImprovedMHM (P: path, k: integer)
4. **IF** $|P| < 2k$ **THEN** // trivial case
5. assign all records to the same cluster (C_1)
6. **RETURN**
7. **END IF**
8. Initialize the directed acyclic graph $M(V, E), |V| = |P| + 1$
9. **FOR** i = 0 **TO** $|P| - k$
10. **FOR** $j = i + k$ **TO** $min\ (|P|, i + 2k - 1)$
11. **IF** i = 0 **AND** $j = i + k$ **THEN**
12. $CurrentCentroid$ = MEAN (X_1 to X_k)
13. $CurrentSSE$ = calculate SSE based on equation (1)
14. $BaseCentroid = CurrentCentroid$
15. $BaseSSE = CurrentSSE$
16. **ELSEIF** $j = i + k$
17. $Delta = X_{i+k} - X_i$
18. $CurrentCentroid = BaseCentroid + Delta/k$
19. $CurrentSSE = BaseSSE + \frac{\sum_{l=1}^{d} Delta[l].\big((1-k)Delta[l] + 2k(X_{i+k}[l] - CurrentCentroid[l])\big)}{k}$
20. $BaseCentroid = CurrentCentroid$
21. $BaseSSE = CurrentSSE$
22. **ELSE**
23. $s = j - i$ // the group size
24. $OldCentroid = CurrentCentroid$
25. $CurrentCentroid = OldCentroid + (X_j - OldCentroid)/s$
26. $CurrentSSE = CurrentSSE + \sum_{l=1}^{d}(X_j[l] - CurrentCentroid[l])(X_j[l] - OldCentroid[l])$
27. **END IF**
28. Draw a directed edge $e = (v_i, v_j)$ and set the weight $w(v_i, v_j) = CurrentSSE$.
29. **END FOR**
30. **END FOR**
31. Compute SP as the shortest path from v_0 to $v_{|P|}$ in $M(V, E)$
32. $ClusterCounter = 1$
33. **FOR EACH** edge $e = (v_i, v_j) \in SP$
34. **FOR EACH** $v_m, i < m \leq j$
35. Assign $X_{P[m]}$ to $G_{ClusterCounter}$
36. **END FOR**
37. $ClusterCounter = ClusterCounter + 1$
38. **END FOR**
39. **RETURN** A
40. **END FUNCTION**

current loop iteration is usually a good starting point for the next iteration which allows a considerable computational saving. In all tests, the $MIN\_IL\_CHANGE = 10^{-7}$ and $MAX\_ITERATION = 100$ are used.

5.1 Tests on standard and synthetic data sets

Experiments were performed on real-world and multiple synthetic benchmark data sets. The results are compared with the results of MDAV. Benchmark data sets which are used in previous studies such as [34, 40, 42, 52] are described in Table 2. All data sets contain numeric attributes without any missing values.

Table 2 Standard benchmark data sets for microaggregation comparison [52]

Data set name	Number of data records (n)	Number of numeric attributes (d)
Tarragona	834	13
Census	1080	13
EIA	4092	11

Table 3 shows the information loss for various values of k, and the elapsed time used for comparison with MDAV method. All elapsed times in second are the av-

Table 3 Information loss and running time comparison for various standard data sets

Data set	k	NPB	$Time_{IMHM}$ (sec)	$Time_{MDAV}$ (sec)	IL_{IMHM}	IL_{MDAV}	*Utility Gain*
Tarragona	3	1	2	<1	16.9305	16.9326	0.01
	4	1	6	<1	19.515	19.5458	0.16
	5	1	4	<1	22.1861	22.4613	1.23
	6	1	1	<1	25.5058	26.3252	3.11
	10	1	1	<1	30.7841	33.1929	7.26
	25	1	1	<1	43.4523	46.9751	7.50
	50	1	<1	<1	56.4309	58.5269	3.58
	100	1	<1	<1	68.1937	69.5501	1.95
Census	3	1	13	<1	5.3668	5.6922	5.72
	4	1	11	<1	6.8577	7.4947	8.50
	5	1	3	<1	8.4165	9.0884	7.39
	6	1	3	<1	9.3118	10.3847	10.33
	10	1	2	<1	12.2284	14.1559	13.62
	25	1	1	<1	18.6134	21.4025	13.03
	50	1	2	<1	25.0804	28.9962	13.50
	100	1	1	<1	32.6775	39.0634	16.35
EIA	3	4	39	4	0.37499	0.4829	22.35
		10	15	4	0.38089		21.12
	4	4	32	3	0.52236	0.6714	22.20
		10	12	3	0.53739		19.96
	5	4	25	2	0.75761	1.6667	54.54
		10	11	2	0.84493		49.31
	6	4	24	2	0.94998	1.3078	27.36
		10	7	2	1.0089		22.86
	10	4	19	1	2.1788	3.8397	43.26
		10	5	1	2.2247		42.06
	25	4	9	<1	7.3893	8.2846	10.81
		10	2	<1	7.5387		9.00
	50	4	5	<1	12.9248	15.1112	14.47
		10	2	<1	13.2641		12.22
	100	1	10	<1	19.7969	20.9065	5.31
		4	3	<1	19.9918		4.38

erage of 10 runs, excluding disk operations for reading the input data set, writing the final protected data set, and (de)normalization. The results of Table 3 are also illustrated in Fig. 5, for comparison of IMHM and MDAV based on the improvement in decreasing the information loss, $Utility\ Gain = (IL_{MDAV} - IL_{IMHM})/(IL_{MDAV}) \times 100\ \%$. In all tests, MDAV is used for initialization without any pre-partitioning.

Additionally, IMHM results on 25 synthetic, uniform, distributed data sets are reported (Table 4). Each data set consists of $n = 100$ records with $d = 10$ numerical attributes drawn from $[-1000, 1000]$ interval by a simple random sampling. Another simulated test set contains 25 synthetic data sets, somehow similar to [42].[13] Each data set has $c = 5$ clusters, each of them contains $n = 20$ multivariate Gaussian distributed records around. All records of a cluster center have a mean value equal to the cluster center and the covariance matrix $\sum_i = \sigma_i^2 I_d$, where σ_i is randomly selected from $[0.1, 0.3]$ and I_d denotes the $d \times d$ identity matrix. Centers are uniformly distributed over the d-dimensional hypercube of $[0, 1]^d$, where d is a random number between 4 and 8. These 2 test sets are named Sim-1 and Sim-2, respectively. The Sim-3 test set is generated in the same manner of Sim-2, except that centers are selected from $[0, 2]^d$, which makes

[13] In the referred paper, number of clusters (c) and records in a cluster (n_i) was chosen randomly.

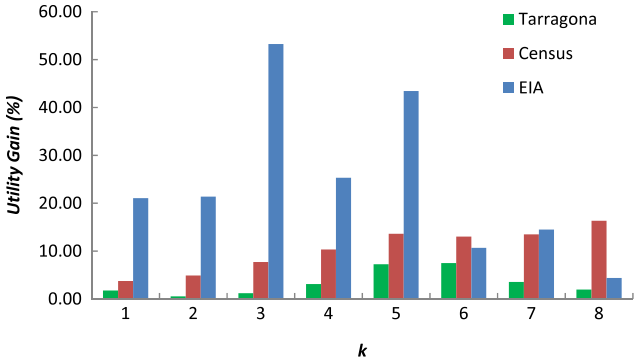


Fig. 5 The Utility Gain of IMHM over MDAV in Table 3, for different values of k on standard real-world data sets ($NPB_{IMHM} = 4$ for EIA)

Table 4 The results of IMHM on uniform and Gaussian synthetic data sets

Data set	k	IL_{IMHM}	IL_{MDAV}	IL_{CBFS}	*Utility* $Gain_{MDAV}$	*Utility* $Gain_{CBFS}$
Sim-1	3	30.4921	32.6804	31.9960	8.73	6.78
	5	43.7689	48.6520	48.0127	10.45	9.25
	10	58.7872	67.5220	66.1611	12.94	11.15
Sim-2	3	13.5109	14.3219	14.1144	8.16	6.81
	5	20.8667	23.5619	23.1056	11.90	10.16
	10	29.8816	37.1805	36.5719	19.63	18.29
Sim-3	3	14.308	15.5112	14.9467	9.93	6.53
	5	21.1656	23.9646	23.3095	12.45	9.99
	10	30.2814	37.7317	36.2494	20.18	16.92

the data sets more clustered. For completeness, we have also reported the results of a slight improved variant of MDAV, i.e., CBFS in Table 4. The average values of utility improvements over MDAV and CBFS methods in terms of *IL* for different values of k are also denoted as *Utility* $Gain_{MDAV}$ and *Utility* $Gain_{CBFS}$, respectively. All runtimes for IMHM, MDAV, and CBFS for a data set are less than one second. Table 3 indicates the superiority of IMHM as k increases.

In addition, two visual results of IMHM for comparison with the results of [41], on two data sets of uniform and Gaussian distribution for $k = 35$ and $k = 1111$ are illustrated in Figs. 6 and 7, respectively. These data sets contain $n = 200$ and $n = 10000$ records of $d = 2$ attributes. Such data sets may be used in real-world scenarios such as data compression and location-based services [41].

5.2 Protecting large data sets

The next test is to show the adaptability of IMHM for multiple scenarios where the data set is too large. To deal with such situations, a simple change in the Number of Pre-partitioned Blocks (*NPB*) is the solution. If *NPB* is increased, the time complexity reduces at the expense of lower utility in terms of *IL*. The first data set for the test is the first 10000 records of the Forest data set from the UCI KDD with

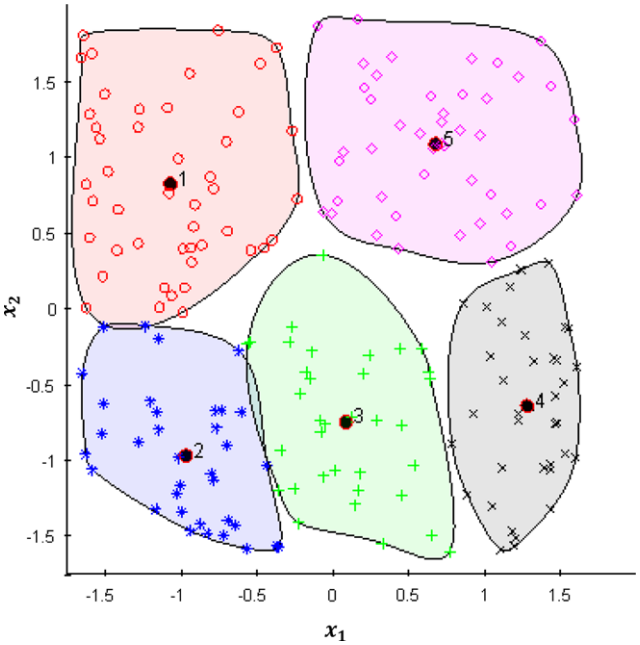


Fig. 6 The result of IMHM on a two-dimensional uniform distributed data set ($n = 200$, $k = 35$). $IL_{IMHM} = 18.8291$, and $IL_{MDAV} = 25.7162$. Small points represent data records in the two-dimensional space (x_1, x_2) along with their numbered clusters

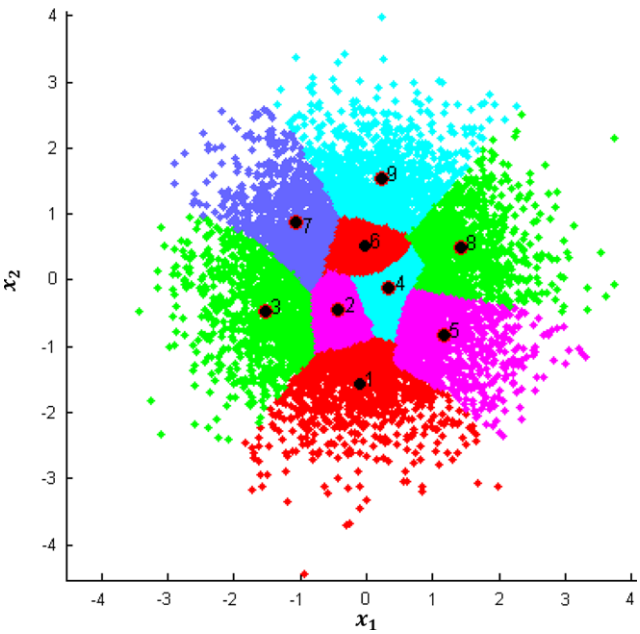


Fig. 7 The result of IMHM on a two-dimensional Gaussian distributed data set ($n = 10000$, $k = 1111$, $NPB = 2$). $IL_{IMHM} = 19.6208$, and $IL_{MDAV} = 22.1158$. Small points represent data records in the two-dimensional space (x_1, x_2) along with their numbered clusters

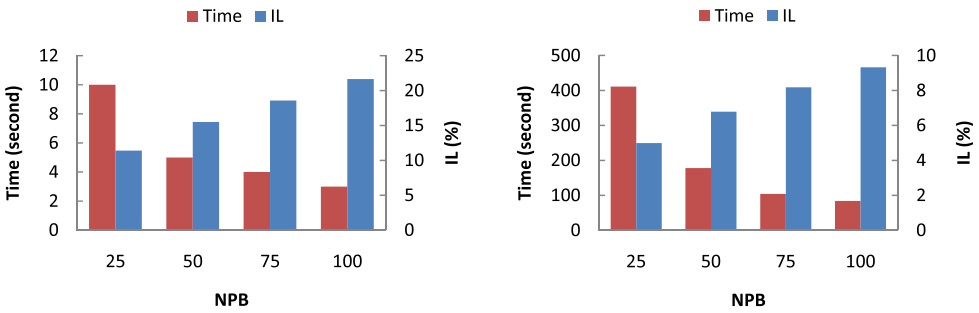


Fig. 8 The impact of the pre-partitioning (NPB) on runtime and IL of IMHM for (*left*) Forest data set and, (*right*) Census Income data set

10 numeric attributes [53], and the next one is the Census Income from UCI repository [54]. The latter data set contains 48842 records with 6 numerical attributes (age, fnlwgt, education-num, capital-gain, capital-loss, and hours-per-week). The test is repeated for $NPB \in \{25, 50, 75, 100\}$ and $k = 10$. To decrease the overhead of the initialization algorithm, MDAV, all records are ordered based on z-score and then every k consecutive records are assigned to a group. Then, the cluster centers are updated based on the assignment. As the results in Fig. 8 confirm, the data publisher can set the proper value of *NPB* in order to get a satisfactory tradeoff between anonymization time and information utility in protected data set.

The results of the test show that IMHM can be easily used without any pre-partitioning requirements for most of the real-world data sets of up to about one thousand records. Additionally, this pre-partitioning is not required when the number of clusters decreases (for example, in location-based services, where k is much larger than the privacy requirement of statistical agencies). For such applications, to our knowledge, the results of IMHM are superior to other results presented in the literature.

5.3 Runtime and convergence speed of IMHM

Regarding the runtime of the algorithm, it is fair to say that MDAV is faster than IMHM. However, in our point of view, this does not restrict the applicability of IMHM in real world scenarios, since reviewing the protection results for assessment of disclosure risk may take more time than the protection algorithm itself, enforcing the total procedure to be offline. Moreover, our tests on the convergence speed of IMHM show that the data publisher can stop the algorithm, while the results are close to the local optimum of a less time critical case with more iterations. Figure 9 shows that the results for Tarragona and Census data set for $k = 5, 10$, converge quickly to a local optimum, and the next iterations can be ignored, if the runtime matters.

5.4 Randomized initialization

Toward proving the applicability of IMHM for real world situations, the next tests try to show that if the publisher is not satisfied with the disclosure risk of the protected data set, he/she is not required to use another algorithm or inject noise to the protected data set. In fact, the k-anonymity model realized by the methods mentioned in this study, do not provide enough protection against some attacks related to the deterministic nature of the procedures. These algorithms always try to cluster the records in groups such that the produced protected data set is useful, while the attacker may simulate the procedure knowing the algorithm parameters and behavior. Such attacks are known as minimality attacks [50]. In order to mitigate the effectiveness of such attacks, the publisher may introduce some randomness within the microaggregation process [55]. In IMHM, the initialization procedure is an interesting place to *inject* some non-deterministic behavior. The IMHM can be initialized with some random centroids and find the optimal assignment for them.

The Randomized IMHM (RIMHM) generates a *semi-random* permutation of all records and calculates the optimal clustering with regard to this permutation, using the ImprovedMHM. If the initialization of the algorithm behaves non-deterministically, repeating the whole anonymization process for an attacker would be difficult. However, this random initialization does not necessarily result in a satisfying information loss, so we have used a semi-random permutation. In order to enable the data publisher, we make a tradeoff between the risk of minimality attack and information loss, we have introduced a parameter, $p \in [0, 1]$ in RIMHM, which controls the amount of the randomness in the initialization step of the algorithm. In order to generate a semi-random permutation, all records are sorted based on z-score. All records in the data set are partitioned into some blocks each containing $n \cdot p$ records (except probably the last block, with more records). Then, all records in a block are randomly permuted and the final permutation is given to ImprovedMHM. The process is somehow similar to rank swapping which is implemented in the μ-ARGUS software [30]. If the parameter increases, more randomness is introduced, because the initial centers are more scattered in the domain space of the data set. The remaining parts of the RIMHM are similar to IMHM. Table 5 shows the average *IL* after 10 executions of RIMHM for the same data sets

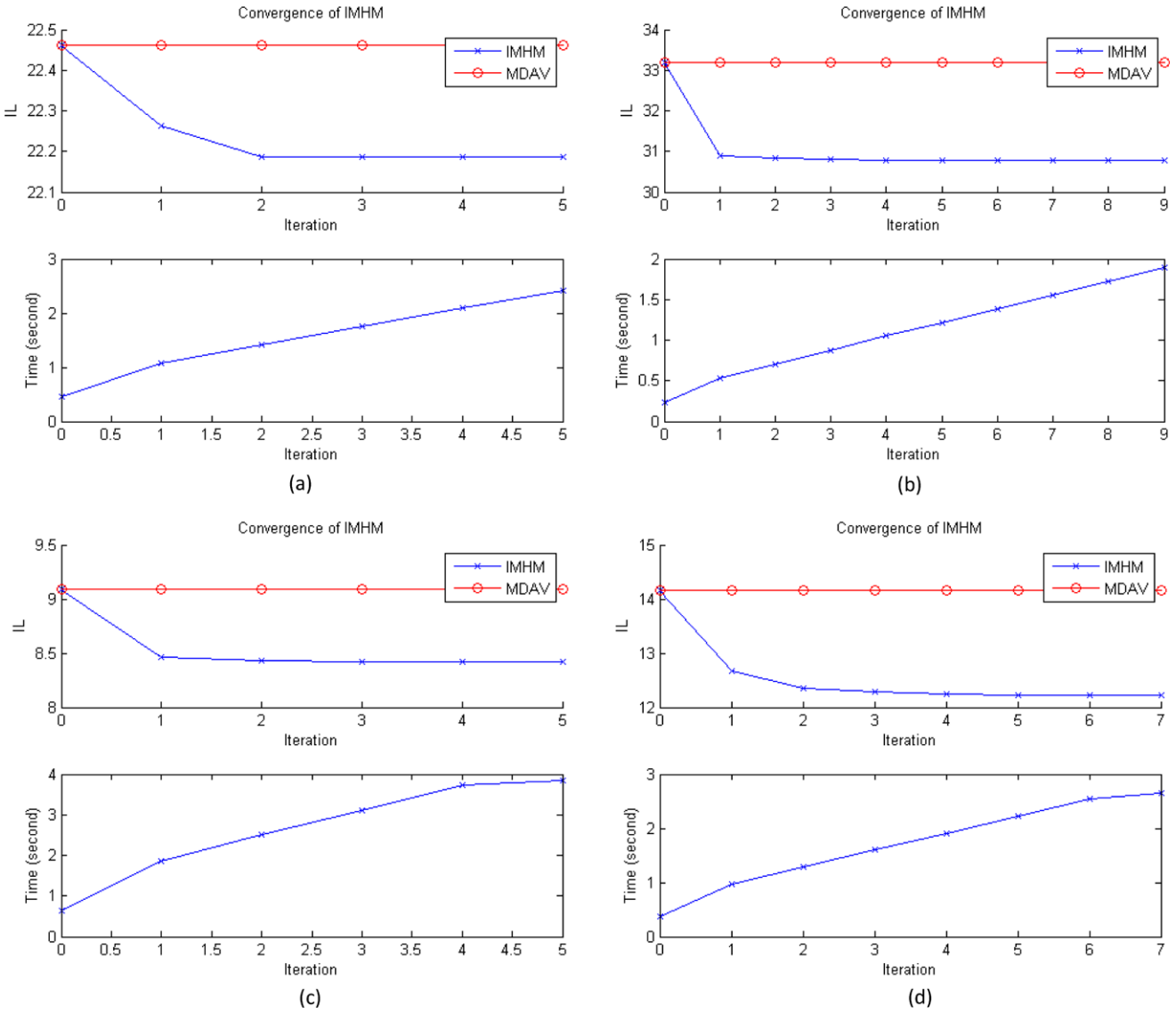


Fig. 9 Convergence speed of IMHM, for (**a**) Tarragona, $k = 5$, (**b**) $k = 10$, and (**c**) Census, $k = 5$, (**d**) $k = 10$. Both *IL* (%) and time (including initialization) after each iteration are presented for each test

and settings of Table 3 ($p = 10$ % is used). All runtimes are for such an execution of RIMHM.

6 Conclusions

This paper presents a novel approach to solving the microaggregation problem in the field of SDC, IMHM. The IMHM forms clusters of records using an iterative optimization method. In each iteration, *SSE* is reduced after calculating the assignment and centroids. The IMHM uses ImprovedMHM to get rid of the local optimum. The comprehensive tests on synthetic and standard data sets for various privacy parameters k, show the superiority of IMHM in terms of data utility measured by *IL* over MDAV. Especially, when the privacy requirement k, increases in some applications such as data compression or location-based services, the clear dominance of IMHM over other methods justifies it as a promising method. However, the IMHM is optimized to decrease the *IL* of its input in an offline manner for the applications where the utility has much more importance. Additionally, the runtimes are comparable to similar recent methods such as PCL [41] and makes it applicable for real-world situations.

The IMHM is a batch method: it accepts all records in the data set at once and calculates the assignment iteratively. Regarding the incremental nature of the algorithm, it is an interesting issue to consider the addition of records to the problem and update the results based on the already found ones. This may be the case where not all records are present

Table 5 Random initialization of IMHM (RIMHM)

Data set	k	NPB	$Time_{RIMHM}$ (sec)	IL_{RIMHM}
Tarragona	3	1	12.55 ± 1.55	16.99 ± 1.59
	4	1	9.66 ± 2.29	20.27 ± 1.17
	5	1	7.84 ± 1.87	24.54 ± 1.00
	6	1	6.94 ± 1.49	25.71 ± 1.44
	10	1	4.43 ± 0.75	32.42 ± 1.22
	25	1	2.32 ± 0.57	43.61 ± 0.16
	50	1	2.22 ± 0.47	56.46 ± 0.15
	100	1	1.56 ± 1.10	68.29 ± 0.14
Census	3	1	15.41 ± 3.51	5.43 ± 0.12
	4	1	15.46 ± 2.90	7.03 ± 0.20
	5	1	11.55 ± 1.09	8.38 ± 0.22
	6	1	11.27 ± 1.52	9.54 ± 0.36
	10	1	8.14 ± 1.45	12.54 ± 0.23
	25	1	4.68 ± 1.49	18.82 ± 0.17
	50	1	1.89 ± 0.53	25.12 ± 0.21
	100	1	1.22 ± 0.13	32.56 ± 0.05
EIA	3	4	41.69 ± 2.15	0.45 ± 0.08
		10	15.78 ± 00.94	0.86 ± 00.24
	4	4	32.18 ± 6.78	0.77 ± 0.44
		10	13.52 ± 03.14	1.39 ± 00.41
	5	4	29.22 ± 1.19	0.87 ± 0.05
		10	10.66 ± 00.65	1.79 ± 00.28
	6	4	27.15 ± 01.33	1.19 ± 00.08
		10	9.74 ± 00.54	2.56 ± 00.39
	10	4	19.74 ± 03.11	2.28 ± 00.19
		10	7.86 ± 02.01	5.49 ± 00.55
	25	4	13.44 ± 4.90	7.50 ± 0.46
		10	6.26 ± 01.30	12.92 ± 00.55
	50	4	12.49 ± 01.59	13.29 ± 00.39
		10	6.17 ± 00.55	21.04 ± 00.95
	100	1	35.76 ± 11.51	20.36 ± 00.71
		4	14.41 ± 01.90	20.61 ± 00.69

at once or in the case of multiple releases. The improvement is considered as a future work in the field of SDC using the microaggregation technique.

Acknowledgement This research is partially supported by ITRC (Iran Telecommunication Research Center) under contract No. 12200/500. We would like to thank the editor and anonymous reviewers for their detailed comments that help improve the paper.

Appendix: Proofs related to the ImprovedMHM function (Fig. 4)

Assume a group of records G with n members $\{X_1, X_2, \ldots, X_n\}$ in a d dimensional space. Let $C_1^n[l] = \sum_{j\in\{1,2,\ldots,n\}} X_j[l]/n$ represents the lth dimension ($1 \leq l \leq d$) of the centroid of G, where $X_j[l]$ denotes the lth dimension of X_j. Based on Eq. (1), it is easy to verify that the SSE of the group can be computed by $SSE(G) = SSE(G[1]) + SSE(G[2]) + \cdots + SSE(G[d])$, where $G[l] = \{X_1[l], X_2[l], \ldots, X_n[l]\}$.

Lemma 1 *If a new record X_{n+1} is added to a group G_1, the SSE of the new group $G_2 = G_1 \cup \{X_{n+1}\}$ denoted by SSE_2 can be calculated based on the calculated SSE before the addition (SSE_1) in $O(1)$.*

Proof Let $\Delta SSE = SSE_2 - SSE_1 = \sum_{1\le l\le d} SSE_2[l] - SSE_1[l]$. Based on Eq. (1), we can expand the $\Delta SSE[l]$:

$$\Delta SSE[l] = \sum_{1\le j\le n+1} \big(X_j[l] - C_1^{n+1}[l]\big)^2 - \sum_{1\le j\le n} \big(X_j[l] - C_1^{n}[l]\big)^2. \quad (3)$$

The new centroid can be computed simply:

$$C_1^{n+1}[l] = \big(n.C_1^n[l] + X_{n+1}[l]\big)/(n+1). \quad (4)$$

So Eq. (3) reduces to:

$$\begin{aligned}\Delta SSE[l] &= \big(X_{n+1}[l] - C_1^{n+1}[l]\big)^2 \\ &\quad + \big(C_1^n[l] - C_1^{n+1}[l]\big)\Big(\sum_{1\le j\le n} X_j[l] - C_1^n[l] \\ &\quad + \sum_{1\le j\le n} X_j[l] - C_1^{n+1}\Big) \\ &= \big(X_{n+1}[l] - C_1^{n+1}[l]\big)^2 \\ &\quad - \big(C_1^n[l] - C_1^{n+1}[l]\big)\big(X_{n+1}[l] - C_1^{n+1}[l]\big) \\ &= \big(X_{n+1}[l] - C_1^{n+1}[l]\big)\big(X_{n+1}[l] - C_1^{n}[l]\big). \quad (5)\end{aligned}$$

These two updates require $O(1)$ computations. □

Equations (4) and (5) are used in the ImprovedMHM function (lines 25–26 of Fig. 4) to update the *SSE* and centroid of a new group.

Lemma 2 *If a record X_{n+1} replaces a member in the group G, say X_1, the SSE of the new group can be calculated in $O(1)$.*

Proof We have $SSE_2[l] = \sum_{2\le j\le n+1}(X_j[l] - C_2^{n+1}[l])^2$. Let $\Delta[l] = X_{n+1}[l] - X_1[l]$, so $C_2^{n+1}[l] = C_1^n[l] + \Delta[l]/n$ and the *SSE* can be updated incrementally:

$$\begin{aligned}&SSE_2[l] - SSE_1[l] \\ &= \sum_{2\le j\le n+1} \big(X_j[l] - C_2^{n+1}[l]\big)^2 - \sum_{1\le j\le n} \big(X_j[l] - C_1^{n}[l]\big)^2 \\ &= \sum_{2\le j\le n} \Big[\big(X_j[l] - C_2^{n+1}[l]\big)^2 - \big(X_j[l] - C_1^{n}[l]\big)^2\Big] \\ &\quad + \big(X_{n+1}[l] - C_2^{n+1}[l]\big)^2 - \big(X_1[l] - C_1^{n}[l]\big)^2 \\ &= \Delta[l]\big((1-n)\Delta[l] + 2n\big(X_{n+1}[l] - C_2^{n+1}[l]\big)\big)/n. \quad (6)\end{aligned}$$

Equation (6) is used in line 19 of Fig. 4, and requires $O(1)$ computations. □

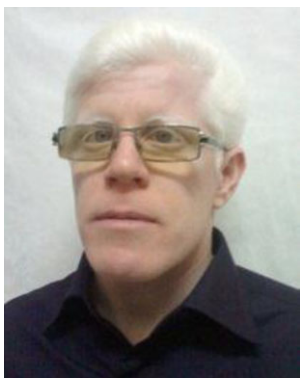
References

1. Sweeney L (2002) K-anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl-Based Syst 10(5):557–570
2. Machanavajjhala A et al (2007) L-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data 1(1):3
3. Ninghui L, Tiancheng L, Venkatasubramanian S (2007) T-closeness: privacy beyond k-anonymity and l-diversity. In: ICDE
4. Hong TP et al (2012) Using TF-IDF to hide sensitive itemsets. Appl Intell 1–9
5. Yin Y et al (2011) Privacy-preserving data mining. In: Data mining. Springer, Berlin
6. Domingo-Ferrer J, Solanas A, Martinez-Balleste A (2006) Privacy in statistical databases: k-anonymity through microaggregation. In: Proceedings of IEEE granular computing, pp 774–777
7. Oganian A, Karr AF (2011) Masking methods that preserve positivity constraints in microdata. J Stat Plan Inference 141(1):31–41
8. Domingo-Ferrer J, Torra V (2001) Disclosure control methods and information loss for microdata. In: Doyle P, Lane JI, Theeuwes JJM, Zayatz L (eds) Confidentiality, disclosure, and data access: theory and practical applications for statistical agencies, pp 93–112
9. Xu C et al (2012) Efficient fuzzy ranking queries in uncertain databases. Appl Artif Intell 37(1):47–59
10. Wong RC-W et al (2006) (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, Philadelphia, pp 754–759
11. Sun X, Li M, Wang H (2011) A family of enhanced-diversity models for privacy preserving data publishing. Future Gener Comput Syst 27(3):348–356
12. Sun X, Sun L, Wang H (2011) Extended k-anonymity models against sensitive attribute disclosure. Comput Commun 34(4):526–535
13. Defays D, Nanopoulos P (1993) Panels of enterprises and confidentiality: the small aggregates method
14. Domingo-Ferrer J, Mateo-Sanz JM (2002) Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans Knowl Data Eng 14(1):189–201
15. Domingo-Ferrer J, Sramka M Disclosure control by computer scientists: an overview and an application of microaggregation to mobility data anonymization
16. Rebollo-Monedero D, Forné J, Soriano M (2011) An algorithm for k-anonymous microaggregation and clustering inspired by the design of distortion-optimized quantizers. Data Knowl Eng
17. Domingo-Ferrer J, Torra V (2005) Ordinal continuous and heterogeneous k-anonymity through microaggregation. Data Min Knowl Discov 11(2):195–212
18. Torra V (2004) In: Domingo-Ferrer J, Torra V (eds) Microaggregation for categorical variables: a median based approach, privacy in statistical databases. Springer, Berlin, pp 162–174
19. Hansen SL, Mukherjee S (2003) A polynomial algorithm for optimal univariate microaggregation. IEEE Trans Knowl Data Eng 15(4):1043–1044
20. Solanas A, Sebé F, Domingo-Ferrer J (2008) Micro-aggregation-based heuristics for p-sensitive k-anonymity: one step beyond. In: Proceedings of the 2008 international workshop on privacy and anonymity in information society. ACM, New York
21. Domingo-Ferrer J, Sebé F, Solanas A (2008) An anonymity model achievable via microaggregation. In: Secure data management, pp 209–218
22. Jian-min H, Ting-ting C, Hui-qun Y (2008) An improved V-MDAV algorithm for l-diversity. In: 2008 international symposiums on information processing (ISIP). IEEE, New York
23. Skinner CJ et al (1994) Disclosure control for census microdata. J Off Stat 10(1):31–51

24. Nin J, Herranz J, Torra V (2008) On the disclosure risk of multivariate microaggregation. Data Knowl Eng 67(3):399–412
25. Yancey W, Winkler W, Creecy R (2002) Disclosure risk assessment in perturbative microdata protection. In: Inference control in statistical databases, pp 49–60
26. Chang C-C, Li Y-C, Huang W-H (2007) TFRP: an efficient microaggregation algorithm for statistical disclosure control. J Syst Softw 80(11):1866–1878
27. Domingo-Ferrer J, González-Nicolás Ú (2010) Hybrid microdata using microaggregation. Inf Sci 180(15):2834–2844
28. Sun X et al (2012) An approximate microaggregation approach for microdata protection. Expert Syst Appl 39(2):2211–2219
29. Crises G (2004) Microaggregation for privacy protection in statistical databases. Rovira i Virgili, Univ. Tarragona, Spain, Tech. Rep. CRIREP-04-005
30. Hundepool A et al (2006) CENEX SDC handbook on statistical disclosure control, version 1.01
31. Oganian A, Domingo-Ferrer J (2001) On the complexity of optimal microaggregation for statistical disclosure control. Stat J United Nations Econ Comission Eur 18(4):345–354
32. Defays D, Nanopoulos P (1993) Panels of enterprises and confidentiality: the small aggregates method. In: Proceedings of the 1992 symposium on design and analysis of longitudinal surveys, Ottawa, Canada
33. Mateo Sanz JM, Domingo i Ferrer J (1998) A comparative study of microaggregation methods. Questiió 22(3):511–526
34. Domingo-Ferrer J et al (2006) Efficient multivariate data-oriented microaggregation. VLDB J 15(4):355–369
35. Heaton B (2012) New record ordering heuristics for multivariate microaggregation. Nova Southeastern University
36. Laszlo M, Mukherjee S (2005) Minimum spanning tree partitioning algorithm for microaggregation. IEEE Trans Knowl Data Eng 17(7):902–911
37. Solanas A, Martinez-Balleste A, Domingo-Ferrer J (2006) V-MDAV: a multivariate microaggregation with variable group size. In: COMPSTAT'2006
38. Chang CC, Li YC, Huang WH (2007) TFRP: an efficient microaggregation algorithm for statistical disclosure control. J Syst Softw 80(11):1866–1878
39. Solanas A et al (2006) Multivariate microaggregation based genetic algorithms. In: 2006 3rd international IEEE conference on intelligent systems
40. Lin JL et al (2010) Density-based microaggregation for statistical disclosure control. Expert Syst Appl 37(4):3256–3263
41. Rebollo-Monedero D, Forné J, Soriano M (2011) An algorithm for k-anonymous microaggregation and clustering inspired by the design of distortion-optimized quantizers. Data Knowl Eng 70(10):892–921
42. Panagiotakis C, Tziritas G (2011) Successive group selection for microaggregation. IEEE Trans Knowl Data Eng. http://www.computer.org/csdl/trans/tk/preprint/ttk2011990169-abs.html
43. Li Y et al (2002) A privacy-enhanced microaggregation method. Foundations of Information and Knowledge Systems, 237–250
44. Solanas A (2008) In: Yang A, Shan Y, Bui L (eds) Success in evolutionary computation. Springer, Berlin, pp 215–237
45. Fayyoumi E, Oommen BJ (2010) A survey on statistical disclosure control and micro-aggregation techniques for secure statistical databases. Softw Pract Exp 40(12):1161–1188
46. MacQueen JB (1967) Some methods for classification and analysis of multivariate observations, California, USA
47. Cormen TH et al (2009) Introduction to algorithms, 3rd edn. MIT Press, Cambridge, p 850
48. Ghouila-Houri A (1962) Caractérisation des matrices totalement unimodulaires. C R Math Acad Sci 254:1192–1194
49. Korte BH, Vygen J (2006) Combinatorial optimization: theory and algorithms. Algorithms and combinatorics, vol 21. Springer, Berlin
50. Wong RCW et al. (2007) Minimality attack in privacy preserving data publishing. In: Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment
51. Mosek AS (2010) The MOSEK optimization software. Online at http://www.mosek.com
52. Brand R, Domingo-Ferrer J, Mateo-Sanz J (2002) Reference data sets to test and compare SDC methods for protection of numerical microdata. European Project IST-2000-25069 CASC. http://neon.vb.cbs.nl/casc
53. UCI KDD archive. Available from: http://kdd.ics.uci.edu/
54. S, H. and B. SD (1999) The UCI KDD archive
55. Cormode G et al (2010) Minimizing minimality and maximizing utility: analyzing method-based attacks on anonymized data. Proc VLDB Endow 3(1–2):1045–1056

Reza Mortazavi is a Ph.D. candidate of computer engineering at Tarbiat Modares University (TMU). He received the M.Sc. degree in Software Engineering from K.N. 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.

Saeed Jalili received the Ph.D. degree from Bradford University (UK) in 1991 and the M.Sc. degree in computer science from Sharif University of Technology in 1985. Since 1992, he has been Associate Professor at Tarbiat Modares University (TMU). His main research interests are Self-* Systems, Software Runtime Verification, Privacy Preserving Data Publishing, and Quantitative Evaluation of Software Architecture.



Hojjat Gohargazi received his M.Sc. degree in Computer Engineering from Tarbiat Modares University, Tehran, Iran, in 2012, and the B.Sc. degree in Computer Engineering from Shahid Bahonar University, Kerman, Iran in 2009. His main research interests are computational intelligence, machine learning and security in Mobile Ad hoc Networks.