The ISC Int'l Journal of Information Security

January 2015, Volume 7, Number 1 (pp. 15–26)

http://www.isecure-journal.org

A Novel Local Search Method for Microaggregation☆

Reza Mortazavi [1,*], and Saeed Jalili [2]

[1] School of Engineering, Damghan University, Damghan, Iran

[2] Computer Engineering Department, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO.

Article history:
Received: 7 January 2015
Revised: 28 June 2015
Accepted: 15 July 2015
Published Online: 18 July 2015

Keywords:
Microaggregation, Privacy Preserving Data Publishing, k-anonymity, Clustering.

ABSTRACT

In this paper, we propose an effective microaggregation algorithm to produce a more useful protected data for publishing. Microaggregation is mapped to a clustering problem with known minimum and maximum group size constraints. In this scheme, the goal is to cluster n records into groups of at least k and at most $2k-1$ records, such that the sum of the within-group squared error (SSE) is minimized. We propose a local search algorithm which iteratively satisfies the constraints of the optimal solution of the problem. The algorithm solves the problem in $O(n^2)$ time. Experimental results on real and synthetic data sets with different distributions demonstrate the effectiveness of the method in producing useful protected data sets.

1 Introduction

Microaggregation is a perturbative approach used in publishing the records of individuals or companies while preserving the privacy of data owners. Microaggregation is also a mechanism to realize the k-anonymity model [1] in Privacy Preserving Data Publishing (PPDP) [2]. Microaggregation techniques are currently being used by many statistical agencies [3]. Microaggregation may be defined as a constraint-clustering problem, where the number of final clusters is not known a priori, but there are minimum and maximum group size constraints achieving privacy and utility in publishing. In microaggregation, each group should contain at least k records. After partitioning records in clusters, the data publisher replaces the records by cluster centers, and these centroids are published. This replacement implies that the quality of published information is degraded. The effectiveness of a microaggregation algorithm is measured by information loss (IL). Lower values of IL indicate less information distortion after microaggregation, so the protected data set is more useful. The measure will be reduced if similar records are grouped. Additionally, it is proved that for a given privacy requirement in terms of k, all groups with more than $2k-1$ records can be split into more groups in order to decrease IL, without violating the privacy requirement [4].

The optimal microaggregation problem can be solved for the univariate case in polynomial time [5], while it has been proved to be NP-hard for multivariate data sets [6]. Many heuristic algorithms have been introduced in the literature, and most of them have some restrictions on the distribution of records in the data set. On the contrary, in this paper, we have proposed an iterative constraint satisfaction method for multivariate microaggregation problem, where a local search in the neighborhood of an initial solution is conducted to find and satisfy some necessary conditions of an optimal solution. Multiple experiments on both real-world and synthetic data sets confirm the effectiveness of the approach to produce useful protected data in terms of IL.

☆ This article is an extended/revised version of an ISCISC'14 paper.

* Corresponding author.

Email addresses: r_mortazavi@du.ac.ir (R. Mortazavi), sjalili@modares.ac.ir (S. Jalili).

ISSN: 2008-2045

The remainder of this paper is organized as follows. Section 2 reviews some related works to solve the problem. Section 3, formalizes the problem, while Section 4 presents the proposed method. Experimental results are given in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

There are multiple heuristics introduced in the literature for solving the microaggregation problem for multivariate data sets [7–11]. These methods are classified into fixed size and data oriented algorithms. In a fixed size algorithm, the number of clusters is fixed to $\lfloor n/k \rfloor$ and exactly k records are aggregated within a cluster (except possibly a few clusters when k does not divide n). Generally, fixed size algorithms are more efficient, but usually result in more distorted protected data. In contrast, data oriented methods do not specify the exact number of group members beforehand, and produce more useful protected data at the expense of time complexity [12].

Domingo-Ferrer and Torra [7] introduced a famous method called Maximum Distance to Average Vector (MDAV). The method repeatedly forms two clusters, each containing k members around records far from the centroid of the data set [1] . It is classified as a fixed size algorithm and is usually referred to for comparison purposes in the literature. Chang *et al.* [9] proposed Two Fixed Reference Points (TFRP). 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 less than k records remain, which are added to their nearest clusters finally. A post-processing step is conducted to refine the clusters. Notably, fixing R_1 and R_2 as reference points makes TFRP an efficient approach with a time complexity of $O(n^2/k)$ that produces lower IL in comparison with MDAV, especially for sparse data sets with a large k [9].

There are also some extensions of optimal univariate microaggregation which are introduced by Hansen and Mukherjee (called MHM) [5] for multivariate cases. Domingo-Ferrer *et al.* [8] proposed a method to order records in multi-dimensional domain space, and form a path based on some heuristics such as Nearest Point Next (NPN), MDAV-MHM, and MD-MHM with quadratic time complexity. Then, an adaptation of MHM is used over the records on the path. Despite the fact of significant reduction of information loss due to these heuristics in some cases, in comparison with recent methods such as [10] and [13], it becomes clear that there is no unique dominating method producing the most useful protected data for different data sets. Laszlo and Mukherjee [11] introduced microaggregation based on the minimum spanning tree with a time complexity of $O(n^2)$. Initially, a complete graph is constructed where each node is a representation of a record in the data set. The weight of an edge is the Euclidean distance between its two endpoints (corresponding to two records in the data set). This method iteratively removes edges in a decreasing order of their associated weights unless this removal results in a component with less than k nodes. Experimental results indicate the method is sensitive to outlier records and may result in more distorted protected data when the minimum distance between optimal groups is low [10]. DBA is an approach based on a density heuristic [13]. 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 to 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 $k-1$ closest neighbors, where p is a tuning parameter provided by the data publisher. Another method is successive Group Selection based on sequential Minimization of SSE (GSMS) [10], with time complexity $O(n^2 \log n)$. The GSMS iteratively discards a candidate cluster which minimizes the current SSE of remaining records [10], and finally assigns remained records to their respective closest clusters. The improved version of GSMS is followed by a post-processing procedure similar to the second phase of TFRP and is called GSMS-T2 [10]. The GSMS-T2 produces comparable information loss regarding other algorithms with similar time complexity. Recently, Laszlo and Mukherjee [14] introduced a local search method and employed it in an iterated local search algorithm. They also repeated their method with 4000 different initial solutions to get the best among different local optimal solutions. In another study, Mortazavi *et al.* proposed a Fast Data oriented Microaggregation algorithm (FDM) [15] that produces k-anonymous versions of a dataset for multiple successive values of k in a single run.

Nevertheless, as the results of the methods confirm, there is still a noticeable diversity between the best

[1] Pseudo-code and more details of MDAV algorithm are presented in Section 4.

[2] The terms “merge” and “noMerge” are used in the original paper.

results of the algorithms for different data sets, which makes them difficult for use in practice. The data publisher may be required to change the anonymization method for different data sets or even for different privacy parameters k. These changes may introduce a number of different tuning parameters that must be set by the data publisher in the new setting, which makes complex the process of anonymization. It is notable to say that microaggregation mechanism is designed originally for numerical data sets; however, there are extensions for other data types [7, 16]. More thorough surveys about microaggregation methods, are presented in [17].

3 Problem Definition

In this section, we formulate the microaggregation problem. Suppose a numerical data set T of n records $x_i, i \in \{1, \ldots, n\}$ in a d dimensional space is given. The data publisher provides an input value k as the privacy parameter. This value is usually lower than 10 in practical Statistical Disclosure Control (SDC) usages. However, it may be increased up to hundreds in some real-world applications, such as Location-Based Services (LBS) [18]. Microaggregation algorithm partitions the data set into c groups in such a way that the following two constraints are satisfied.

(1) The privacy and utility constraints are satisfied, i.e., $k \leq |G_p| < 2k$, $p \in \{1, \ldots, c\}$, where $|G_p|$ denotes the number of records in G_p.
(2) The whole data set is partitioned into c non-overlapping groups, i.e., $\cup_{p=1}^{c} G_p = T$, and $G_p \cap G_q = \emptyset, \forall p, q \in \{1, \ldots, c\}, p \neq q$.

If a given microaggregation algorithm satisfies the above constraints, its solution is called a valid solution. The sum of within-group squared error (SSE) is minimized to make the records in a group more similar. This measure is formulated in Equation 1.

$$SSE = \sum_{p=1}^{c} \sum_{j=1}^{|G_p|} (x_{pj} - \bar{x}_p)^T (x_{pj} - \bar{x}_p) \qquad (1)$$

where x_{pj} is the j-th record of G_p and $\bar{x}_p$ denotes the centroid of G_p, $\bar{x}_p = 1/|G_p| \sum_{j=1}^{|G_p|} x_{pj}$. The value of SSE is usually normalized by SST, calculated as in Equation 2, where $\bar{x}$ is the average of the whole data set.

$$SST = \sum_{p=1}^{c} \sum_{j=1}^{|G_p|} (x_{pj} - \bar{x})^T (x_{pj} - \bar{x}). \qquad (2)$$

The normalized measure $IL = SSE/SST * 100\%$ is always between 0% and 100%. Lower values of IL indicate more similar centroids to original records and less quality degradation due to the perturbation. In this paper, the IL measure is used to quantify and compare the performance of the proposed method.

The result of a microaggregation algorithm may be considered as an assignment of records to groups, where the membership of a record x_{pi} to a group G_p is denoted by $x_{pi} \to G_p$. The cost of this assignment is a partial summand of SSE, and may be denoted by $PSSE(x_{pi} \to G_p)$.[3] In other words, $PSSE(x_{pi} \to G_p) = (x_{pi} - \bar{x}_p)^T (x_{pi} - \bar{x}_p)$, denotes the increased error related to assigning x_{pi} to G_p in comparison with the error before the assignment. The final value of SSE can be calculated by summing these partial summands, i.e., $SSE = \sum_{p=1}^{c} \sum_{j=1}^{|G_p|} PSSE(x_{pj} \to G_p)$. Its notable that changing the assignment of records among some clusters has no effect on the $PSSE$ of other clusters.

4 The Proposed Algorithm

In this section, the Iterative Constraint Satisfaction for Microaggregation (ICSM) is presented. Section 4.1 defines the exchange and migration constraints and then introduces the ICSM algorithm. Section 4.2 introduces the Tour2Assignment method to determine the number of clusters, c. Section 4.3 analyses the algorithm.

4.1 Algorithm Description

In this section, we introduce our method for microaggregation problem. Before describing the algorithm, we define some constraints of an optimal partitioning in the microaggregation problem. In addition to cluster size constraints, other constraints are necessary to be satisfied in an optimal solution, S_{opt}. Suppose two records $x_{pi} \in G_p$ and $x_{qj} \in G_q$ such that $p, q \in \{1, \ldots, c\}, p \neq q$ in an optimal solution S_{opt} are given. The optimality of S_{opt} guarantees that exchanging the assigned clusters of these records, G_p and G_q, without any other change in S_{opt}, will not reduce the information loss in terms of IL. This constraint is formalized in definition 1.

Definition 1. Exchange constraint: Let $x_1 = x_{pi}$, $x_2 = x_{qj}$, $SSE_1 = PSSE(x_1 \to G_p) + PSSE(x_2 \to G_q)$, and $SSE_2 = PSSE(x_1 \to G_q) + PSSE(x_2 \to G_p)$. In an optimal solution of a microaggregation problem, the quantity of $dSSE = SSE_2 - SSE_1$ should be always non-negative, otherwise, we can exchange the assignment of these records and gain more infor-

[3] We assume that the subscripts denote the same entities, independent of the new assignment. For example, G_p indicates the same cluster before and after the assignment. However, it is obvious that any change in the assignment will also change the exact values of corresponding centroids, and the subscripts do not show them.

mation utility. In this paper, this constraint is called exchange constraint.

Another constraint in an optimal solution of a microaggregation problem is related to migration of a record to another cluster, i.e., changing the assignment of a record from a more crowded cluster to another one. This migration should not result in more utility for protected data. This constraint is formalized in definition 2.

Definition 2. Migration constraint: Let $x_1 = x_{pi}$, $\forall i \in \{1, \ldots, |G_p|\}, p, q \in \{1, \ldots, c\}$, $p \neq q, |G_p| > k$, $|G_q| < 2k - 1$ [4], and $dSSE = PSSE(x_1 \rightarrow G_q) - PSSE(x_1 \rightarrow G_p)$. In an optimal solution S_{opt}, the constraint of $dSSE \geq 0$ as a direct result of the optimality of S_{opt}, is always satisfied. This constraint is called migration constraint in this paper.

Definition 3. n-opt heuristic: Let A be an assignment of records to clusters. Additionally, assume A' denotes another assignment of the same records to the same number of clusters after applying a heuristic H. If H involves exactly $n \geq 1$ change(s) in A to produce A', it is called an n-opt heuristic. More specifically, for $n = 1$ and $n = 2$, H is called 1-opt and 2-opt heuristic, respectively. The ICSM iteratively improves a given solution to produce a more useful protected data set. In this paper, we have considered the output of MDAV [7] for initialization. [5] The pseudo-code of MDAV is presented in Algorithm 1, where its output is an assignment of records to clusters. 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 (lines 2 and 3). The next step is to build two clusters, including r and s and their $k-1$ nearest records, respectively (line 4). These two clusters are microaggregated and removed from the data set. These steps are repeated until less than $2k$ records remain (line 5). Finally, if at least k records remain, a cluster containing all of these unassigned records is formed (line 6). If less than k records remain, they will be added to their nearest clusters (line 7). This method runs in $O(n^2)$ time.

The pseudo-code of ICSM is given in Algorithm 2. The input of ICSM is the normalized data set T of n records each containing d numerical attributes, and a privacy parameter k, while the output is the assignments of records to clusters, A. The ICSM starts from the solution of MDAV (line 2) to initialize A, and converts it to a path P to determine the number of clusters [6] (lines 5 and 6). The procedure continues to improve IL iteratively by satisfying the exchange and migration constraints. If the exchange and/or migration constraints are not satisfied, there is a place to improve IL, which is done by another function, SimpleHeuristic (line 7). If no such simple improvements are found, another function is called to apply a more complicated heuristic (line 9). The algorithm finishes if no such (simple and complex) improvements are found or $MAX\_ITERATIONS$ is reached. Finally, the results are returned in line 11.

Algorithm 1 Pseudo-code of MDAV

Input: T (data set), k (privacy parameter)
Output: A (Assignment of records to clusters)
1: **Function** ([A]=MDAV (T: data set, k: integer))
2: Compute the centroid $\bar{x}$ of whole data set.
3: Find the most distant record r from $\bar{x}$. Also find the most distant record s from r.
4: Form a cluster containing r and its $k-1$ nearest neighbors. From another cluster containing s and its $k-1$ nearest neighbors. Set aside these clusters from the data set.
5: If there are at least $2k$ records remaining (unassigned), repeat steps 2, 3, 4.
6: 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.
7: If there exist at most $k-1$ unassigned records, assign each of them to the closest cluster.
8: Return the assignment of records to clusters.
9: **end Function**

Algorithm 2 Pseudo-code of ICSM

Input: data set T , privacy parameter k
Output: assignment of records to clusters A
1: **Function** ([A]=ICSM (T: data set, k: integer))
2: A = MDAV(T,k) //Algorithm 1
3: **repeat**//loop for n-opt heuristic
4: **repeat**//inner loop for 1-opt and 2-opt heuristics
5: Convert A to a tour (path) P containing all records based on MDAV-MHM (Section 2 and [8])
6: A = Tour2Assignment(P, k) // Determining number of clusters (Section 4.2, **??**)
7: A = SimpleHeuristic(T, k, A) //1-opt and 2-opt heuristics (Algorithm 3)
8: **until** no change in A **OR** MAX_ITERATIONS is reached
9: A = ComplexHeuristic(T, k, A) //n-opt heuristic (Algorithm 4)
10: **until** no change in A **OR** MAX_ITERATIONS is reached
11: **return** A
12: **end Function**

In order to find simple improvements in SimpleHeuristic (Algorithm 3), we construct a directed graph based on the solution of a microaggregation problem for n records and c clusters (lines 2 and 3). Let $H = (V, E)$ be a complete weighted directed graph with $|V| = n + c$ nodes and, $|E| = (n + c)(n + c - 1)$ edges. Each node $v \in V$ represents a record or a cluster of the solution. The weights of all edges connecting two clusters [7] are infinite.

[4] All group sizes are considered before the migration. These conditions are necessary to produce a valid protected data after migration that satisfies the privacy requirement.
[5] See also Section 5.3.
[6] Please see Section 4.2 for more details.
[7] We use the term cluster here to denote a node in H that

Algorithm 3 Pseudo-code of SimpleHeuristic

Input: data set T, privacy parameter k, assignments of records to cluster A, to be improved
Output: improved assignments of records to clusters A

```
1: Function ([A]=SimpleHeuristic (T: data set, k: integer, A: array of assignments))
2:     n = |T| //number of records
3:     c = MAX(A) //number of clusters
4:     Construct/Update a complete weighted directed graph H(V, E) with |V| = n + c nodes. Calculate the weight of all edges in E based on Equation 3, 5, and 6.
5:     Pool = InitializePool(p_n)
6:     for each v1, v2 ∈ V, v1 ≠ v2 do
7:         if w(v1, v2) + w(v2, v1) < 0 AND Pool.Count < p_n then
8:             if v1 OR v2 represents a cluster then
9:                 Pool.Add(Migrate(v1, v2)) //1-opt heuristic
10:            else
11:                Pool.Add(Exchange(v1, v2)) //2-opt heuristic
12:            end if
13:        end if
14:    end for
15:    if Pool.IsEmpty() then return A
16:    end if
17:    Pool.sort() // all elements are sorted such that the best change is placed at the first location.
18:    ChangedClusters= ∅
19:    for each Change ∈ Pool.elements do // in order
20:        ChangingClusters= involved clusters in Change
21:        if ChangingClusters then ∩ ChangedClusters= ∅
22:            Apply Change on A to improve IL
23:            ChangedClusters = ChangedClusters ∪ ChangingClusters
24:        end if
25:    end for
26:    return A
27: end Function
```

There is a directed edge $e \in E$ between any two records[8] (line 7). The weight of such an edge from $x_1 = x_{pi} \in G_p$ to $x_2 = x_{qj} \in G_q$,$q \neq p$ represents the utility gain of replacing the assignment of $x_2 \to G_q$ by $x_1 \to G_q$ without any change to the rest of the assignments (i.e., both $x_1 \to G_p$ and $x_1 \to G_q$ are present in the assignment at the same time, while x_2 is not assigned, temporarily). All records in the same cluster are connected via infinite-weight edges. The weight is denoted by $R_1(x_{pi}, G_q, x_qj)$. Regarding formulae of SSE and PSSE, for any $x_{pi} \in G_p$, and $x_{qj} \in G_q$, we can calculate R_1 function by Equation 3.

$$R_1(x_{pi}, G_q, x_{qj}) = \begin{cases} +\infty & \text{if } p = q \\ \sum_{l=1}^{d} (x_{pi}[l] - x_{qj}[l])2(x_{qj}[l] - \bar{x}_q[l]) & \\ \quad +(x_{pi}[l] - x_{qj}[l])(|G_q| - 1/|G_q|) & \text{if } p \neq q \end{cases} \tag{3}$$

In Equation 3, $x[l]$ denotes the l^{th} element (or attribute) of vector x. Remember that $\bar{x}_q$ is the centroid of G_q and d is the dimension of the underlying data set.

Example 1: Assume a $d = 1$ dimensional data set $T = [2, 3, 4, 5, 6, 7]$ is given for $k = 3$ anonymity. Let $A = [1, 1, 1, 2, 2, 2]$, so $G_1 = [2, 3, 4]$ and $G_2 = [5, 6, 7]$. Therefore, $\bar{x}_1 = 3$, $\bar{x}_2 = 6$ and $SSE = SSE_{G_1)} + SSE_{G_2} = (1 + 0 + 1) + (1 + 0 + 1) = 4$. Based on Equation 3, $R_1(x_2, G_2, x_5) = R_1(3, G_2, 6) = (-3)(2(0) + (-3)2/3) = 6$. This means if $6 \to G_2$ is replaced by $3 \to G_2$ then $G_2 = [5, 3, 7]$ and SSE_{G_2} increases by 6. This is true, since $8 - 2 = 6$.

The value of $dSSE$, introduced in definition 1, is independent of the SSE of all clusters except G_p and G_q, and can be rewritten in terms of the R_1 function by Equation 4.

$$dSSE = SSE_2 - SSE_1 = R_1(x_{pi}, G_q, x_{qj}) + R_1(x_{qj}, G_p, x_{pi}) \tag{4}$$

This equation confirms that an exchange is in fact a combination of two such migrations.[9] The graphical representation of this exchange in graph H, forms a cycle of length two, where each edge is weighted by R_1, as depicted in Figure 1. Finding and eliminating such cycles of length 2 in the graph H, is an effective way to reduce IL, and based on definition 3 is referred to as 2-opt heuristic, since it changes two assignments (line 11). These replacements can also be shown in the form of changing the state of cluster assignments from an initial state of $\{x_1 \to G_p, x_2 \to G_q\}$ to the state of $\{x_1 \to G_q, x_2 \to G_p\}$.

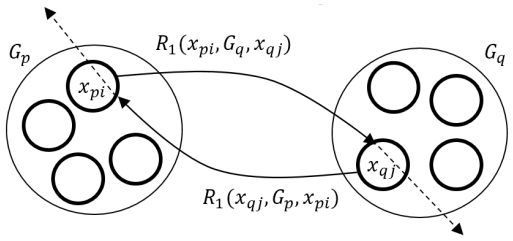

Figure 1. Graphical representation of 2-opt heuristic. After the exchange, x_1 and x_2 are assigned to G_q and G_p, respectively. For clarity, only involved nodes and edges are drawn.

Moreover, migration of a record to a new cluster can be shown as a cycle of length 2, in our graph representation. Let $R_2(x_{pi}, G_q)$ denotes the change in SSE after assigning of x_{pi} to G_q (again, the last assignment of x_{pi} to G_p is preserved, temporarily). This function returns no gain, if G_q is overloaded after migration, which is denoted by infinity in Equation 5 for R_2.

represents a cluster in the solution.

[8] We use the term record here to denote a node in H that represents a record in the data set.

[9] We require R_1 to form more complex improvements, so we do not consider the combined form.

$$R_2(x_{pi}, G_q) = \begin{cases} +\infty & \text{if } p = q \text{ or } |G_q| \geq 2k-1 \text{ (before migration)} \\ \sum_{l \in \{1,\dots,d\}} \frac{|G_q|.(\bar{x}_q[l] - x_{pi}[l])^2}{|G_q|+1} & \text{otherwise} \end{cases} \quad (5)$$

We can form a cycle in H representing the migration of a record to a new cluster, using a new virtual edge from the destination cluster G_q to the source cluster G_p. The weight of this edge is the difference in SSE after removing the source record of migration, x_{pi} from G_p, provided that the source cluster contains enough records after migration. The weight may be calculated by another function $R_3(G_q, x_{pi})$ by Equation 6.

$$R_3(G_p, x_{pi}) = \begin{cases} +\infty & |G_p| \leq k \text{ (before migration)} \\ \sum_{l \in \{1,\dots,d\}} \frac{-|G_p|.(\bar{x}_p[l] - x_{pi}[l])^2}{|G_p|-1} & \text{otherwise} \end{cases} \quad (6)$$

The illustration of the cycle for this migration in terms of R_2 and R_3 is also presented in Figure 2. Based on definition 3, we call this improvement as the 1-opt heuristic, because involves a single change in the assignments.

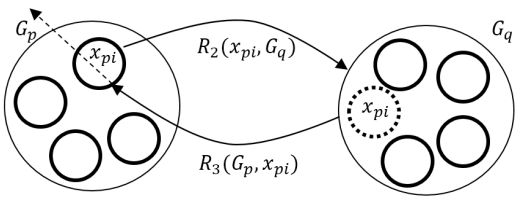

Figure 2. Cyclic representation of migration of x_{pi} from G_p to G_q

The SimpleHeuristic function initializes an empty pool of a predefined size of p_n (line 5). If such cycles of length 2 with negative weights are found, they are added to the pool (lines 6-14). If no such cycles are found, the assignment A is returned without any change (line 15). The SimpleHeuristic function applies the best changes in the $Pool$ in a greedy manner [10] (lines 16-24) and finally returns the improved assignment (line 25).

The migration and exchange constraints may also be generalized to cycles involving more than two clusters, which are done by the ComplexHeuristic function (line 9 of Algorithm 2). Generally, in an optimal solution, there is not any utility gain in terms of IL by a cyclic shift of records from different clusters over their assigned clusters. In other words, there does not exist a valid negative weight cycle in the graph H, corresponding to a series of migrations and/or exchanges. However, finding such negative cost cycles is not a trivial task, because there may be an exponential number of such cycles that must be examined. Additionally, many negative weight cycles in H do not represent a valid cyclic shift of records over clusters, i.e., all negative weight cycles in H, do not represent necessarily an improvement. Figure 3 shows such an invalid cycle in H, where all weights are assumed to be negative.

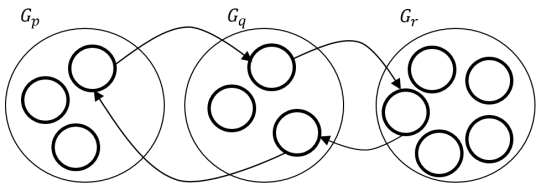

Figure 3. A negative weight cycle that does not (necessarily) result in an improvement of IL. All edge weights are negative. This cycle is invalid, because it involves a cluster (G_q) more than once.

In order to solve these problems, we have removed some edges from H to get a directed acyclic graph $H'(V, E')$, where $E' \subset E$. This transform is done in a heuristic manner to decrease the computation cost of finding one such cycle. Additionally, any negative cost cycle extracted from H', represents a valid cyclic shift of records over clusters, which results in less information loss. However, these benefits are at the expense of losing some possible improvements.

The construction of H' in Algorithm 4 is as follows. First, for each cluster G_p, a value called $Score(G_p)$ is calculated (line 4) as the sum of the weights of all edges from their records minus the sum of the weights of all edges to their records (Equation 7). Intuitively, lower values of $Score(G_p)$ indicate more probability for G_p as a cluster participating in a gainful cyclic shift.

$$Score(G_p) = \sum_{\substack{q \in \{1,\dots,c\}, q \neq p \\ i \in \{1,\dots,|G_p|\} \\ j \in \{1,\dots,|G_q|\}}} R_1(x_{pi}, G_q, x_{qj}) - \sum_{\substack{q \in \{1,\dots,c\}, q \neq p \\ i \in \{1,\dots,|G_p|\} \\ j \in \{1,\dots,|G_q|\}}} R_1(x_{qj}, G_p, x_{pi}) \quad p \in \{1,\dots,c\} \quad (7)$$

Let's specify the cluster with the lowest value of $Score$ as G_s (line 5). Then all cluster centroids are ordered based on their distances to G_s (line 6). The rank of G_p in the ordered list is denoted by $Rank(p)$. All records $x_{pi} \in G_p$, $i \in \{1, \dots, |G_p|\}$ have the same rank as their assigned cluster G_p. All edges $E' \subset E$ in H' are drawn from the records or clusters with lower ranks to the records or clusters with greater ranks (line 7). All other edges are removed (denoted by infinite edge weights). The weights of edges in E' are

[10] Two simultaneous improvements are consistent, if they do not involve a cluster more than once, so applying them at the same time does not result in any conflict. The SimpleHeuristic function applies all changes saved in $Pool$ based on the decreasing order of their impact on IL (line 16), provided that they are consistent (line 20).

Algorithm 4 Pseudo-code of ComplexHeuristic

Input: data set T, privacy parameter k, assignments of records to cluster A, to be improved
Output: improved assignments of records to clusters A

1: **Function** ([A]=ComplexHeuristic (T: data set, k: integer, A: array of assignments))
2: n = $|T|$ //number of records
3: c = *MAX(A)* //number of clusters
4: Calculate $Score(G_p)$, $p \in \{1,\dots,c\}$ based on Equation 7.
5: G_s = The group with minimum *Score*
6: Calculate *Rank(p)*, $p \in \{1,\dots,c\}$ based on the G_p from G_s (closer groups have lower *ranks*)
7: Construct $H'(V,E')$ from $H(V,E)$ and consider updated weights $w_{i,j}$ based on Equation 8.
8: **for each** $v_s \in V$ such that $A[v_s] = G_s$ or $v_s \equiv G_s$ **do**
9: **for each** $v_e \in V, Rank(v_e) \geq Rank(v_s)$ **do**
10: SP = shortest path from v_s to v_e
11: $cost(v_s, v_e)$ = Weight of SP
12: **if** $cost(v_s, v_e) + w_{v_e,v_s} < 0$ **then**
13: Apply all changes mentioned in SP and update A //n-opt heuristic
14: **end if**
15: **end for**
16: **end for**
17: **return** A
18: **end Function**

summarized in Equation 8 [11] for $x_{pi} \in G_p$, $x_{qj} \in G_q$, and $p, q \in \{1,\dots,c\}, p \neq q$.

$$
\begin{aligned}
w_{x_{pi},x_{qj}} &= \begin{cases} R_1(x_{pi}, G_q, x_{qj}) & rank(p) < rank(q) \\ +\infty & rank(p) > rank(q) \end{cases} \\
w_{x_{pi},G_q} &= \begin{cases} R_2(x_{pi}, G_q) & rank(p) < rank(q) \\ +\infty & rank(p) > rank(q) \end{cases} \\
w_{G_p,x_{qj}} &= \begin{cases} R_3(G_q, x_{qj}) & rank(p) < rank(q) \\ +\infty & rank(p) > rank(q) \end{cases} \\
w_{G_p,G_q} &= +\infty
\end{aligned}
\tag{8}
$$

In order to find a valid negative cycle in H, first we calculate the shortest path between all nodes in H' without any incoming edges (i.e., record members of G_s along with the node representing G_s itself, which is denoted by $v_s \equiv G_s$), to the last nodes without any outgoing edges (i.e., record members of the most distant cluster from G_s along with a node representing the most distant cluster itself). These computations are done in lines 8-16. During the shortest path algorithm, if the cost of a middle (or final) node as the distance from some nodes in G_s is changed, we can check whether a returning edge may form a negative weight cycle (line 12). For example, if $cost(v_s, v_e)$ denotes the weight of the shortest path from $v_s \in G_s$ to a node v_e, which represents a record and $Rank(v_e) \geq Rank(v_s)$, we can calculate the cost of the cycle by $cost(v_s, v_e) + R_1(v_e, G_s, v_s)$. Other cycles involving the cluster nodes may also be calculated in the same manner. Note that all such cycles with negative weights extracted in this manner are valid, since there are no two nodes from the same cluster within the shortest paths, otherwise the cost of such paths would be infinite based on the weights calculated by Equation 8. After finding such negative weight cycles, we can eliminate them, and then update the assignments (line 13). Line 17 returns the improved assignments A. The improvements of such cycles may include multiple exchanges or migrations, and are called n-opt heuristics in this paper. As a simple illustrative example, Figure 4 shows a cycle of length 3, involving two records $x_1 \in G_1$, $x_2 \in G_2$, and a cluster G_3. If the weight of the cycle is negative, we can change the assignments from an initial state of $\{x_1 \to G_1, x_2 \to G_2\}$ to the state of $\{x_1 \to G_2, x_2 \to G_3\}$ to decrease IL.

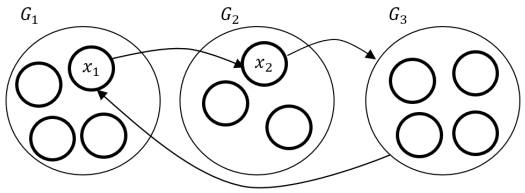

Figure 4. A negative weight cycle that represents n-opt heuristic to improve IL. The improvement consists of replacing x_2 by x_1 in G_2, and migration of x_2 to G_3.

It is notable that applying 1-opt and 2-opt heuristics is more efficient since they involve only two edges. Therefore, these simple heuristics are prioritized as the inner loop (line 4 of Algorithm 2).

4.2 Determination of the Number of Clusters

In data oriented microaggregation algorithms, optimal number of clusters is unknown. We have used a procedure like MDAV-MHM [8], to map the solution to a univariate microaggregation problem, and then use the optimal univariate solution [5] to determine the number of clusters. Fortunately, this procedure may be seen as an improving method, because it is guaranteed to produce a clustering as good as its input [19]. [12] Mapping the assignment to a path is done in line 5 of Algorithm 4. To this end, a cluster containing the most distant record from the centroid of the whole data set is selected. Then, all records in the group are stitched together on a path. Next groups are selected based on the nearest point next (NPN) heuristic such that similar records are closer on the path. This procedure continues until all records are ordered in the path

[11] Ties are broken randomly.

[12] In fact, based on our experiments, in most cases it improves the IL of microaggregation slightly, especially in the initial steps of algorithm execution.

[8]. We have introduced a variant of MHM [5] called Tour2Assignment, to determine the optimal clustering regarding the order of records in the path (line 6 of Algorithm 2). In the original MHM algorithm [5], the shortest path starting from a dummy node to the last node is considered, so the first k records in the path are always assigned to the same cluster. However, despite the ImprovedMHM introduced in [19], Tour2Assignment interprets the input path as a tour of nodes by considering the first node in the path after the last one. This provides the chance for the last nodes in the path to be assigned to the same cluster of the first nodes in the path, potentially leading to lower information loss of the protected data set. [13]

The pseudo-code of Tour2Assignment is presented in **??**. This function groups all records in the same cluster for trivial cases (lines 2-5). [14] A directed acyclic graph $M(V, E)$ with $|V| = |P| + 2k - 1$ nodes is initialized (line 6). Each node v_i corresponds to a record in the path (line 7). The Tour2Assignment inserts an edge starting from v_i to v_{j+i} where $k \leq j < 2k$. The edge $e = (v_i, v_{j+i})$ represents a cluster containing all records v_m, $i < m \leq j + i$ in the path. The weight of an edge is equal to the SSE of respective cluster (lines 10 and 11). However, despite the original MHM algorithm [5], we have implemented an incremental computation of centroids and weights (lines 13-26) similar to [19], which makes it possible to calculate all centroids and weights based on the previous calculated centroids and SSE values (except the first centroid and SSE in lines 10 and 11). This will improve the performance of graph construction significantly. The Tour2Assignment places the first $2k - 1$ starting records in the original path at the end of the path, one after another, and runs the MHM. However, to increase the performance, instead of copying the first records one by one and running the MHM, all common parts of these rotated paths are constructed once, and only the shortest path algorithm for directed acyclic graphs is executed $2k - 1$ times (line 28). Since the optimal shortest path goes through at least one of the consecutive $2k - 1$ nodes in a tour [15], the algorithm finds the optimal clustering after $2k - 1$ iterations. In each iteration, the shortest path from one of the first $2k - 1$ nodes, v_s, in the graph to its copy, v_e, is calculated and the minimum weight path among these shortest paths is selected (lines 29 and 30). Finally, all records covered by an edge are assigned to a new cluster (lines 31-37). The assignment is done in a way that minimizes the changes of cluster numbers, i.e., if the output assignments of the function, say A_1, are converted to a tour again, and the tour is passed to the function to produce another output, say A_2, two outputs will be the same: $A_1 = A_2$. This simplifies stopping condition checking for the inner loop of ICSM (line 8 in Algorithm 2).

Algorithm 5 Pseudo-code of Tour2Assignment function

1: **Function** ([A]=Tour2Assignment (P: tour, k: integer))
2: **if** $|P| < 2k$ **then** // trivial case
3: assign all records to the same cluster (G_1)
4: **return** A
5: **end if**
6: Initialize the directed acyclic graph $M(V, E), |V| = |P| + 2k - 1$
7: $\forall i \in \{1, \dots |P|+2k-1\}, v_i = P[((i-1)$ **MOD** $|P|)+1]$
8: **for** $i = 1$ **TO** P **do**
9: **if** $i = 1$ **then**
10: $CurrentCentriod$ = MEAN(x_{v_2} to $x_{v_{k+1}}$)
11: $CurrentSSE$ = calculate SSE based on Equation 1
12: **else**
13: Delta = $x_{v_{k+i}}$ - x_{v_i}
14: $CurrentCentriod = BaseCentroid$ + Delta/k
15: $CurrentSSE = BaseSSE + \frac{\sum_{l=1}^{d} Delta[l].\left((1-k)Delta[l]+2k(x_{v_{k+i}}[l]-CurrentCentriod[l])\right)}{k}$
16: **end if**
17: $BaseCentroid = CurrentCentriod$
18: $BaseSSE = CurrentSSE$
19: Draw a directed edge $e = (v_i, v_{k+i})$ and set the weight $w(v_i, v_{k+i}) = CurrentSSE$
20: $k_1 = k$
21: **for** $j = i + k + 1$ **TO** $i + 2k - 1$ **do**
22: $CurrentSSE = CurrentSSE + k_1/(k_1 + 1).\sum_{l=1}^{d}(CurrentCentriod[l] - x_j[l])^2$
23: Draw a direct edge $e = (v_i, v_j)$ and set the weight $w(v_i, v_j) = CurrentSSE$
24: $CurrentCentriod = CurrentCentriod + (x_j - CurrentCentriod)/(k_1+1)$ $k_1 = k_1 + 1$
25: **end for**
26: **end for**
27: Compute $SP(s)$ = shortest path from v_s to $v_e, s \in \{1, \dots 2k - 1\}, e = s + |P|$
28: $S = \text{argmin}_s(\text{weight}(SP(s))), s \in \{1 \dots, 2k - 1\}$
29: $SelectedShortestPath = SP(S)$
30: $ClusterCounter = 1$
31: **for each** edge $e = (v_i, v_j) \in SelectedShortestPath$ **do**
32: **for each** $v_m, i < m \leq j$ **do**
33: $A\left[P[\left((m-1)\mathbf{MOD}|P|\right)+1]\right] = ClusterCounter$ //assignment
34: **end for**
35: $ClusterCounter = ClusterCounter + 1$
36: **end for**
37: **end Function**

4.3 Analysis of the ICSM

Generation of initial solution using MDAV requires $O(n^2/k)$ time [7]. The conversion of the assignment to a path involves $O(n^2/k)$ operations [2]. Moreover, determining the number of clusters in Tour2Assignment requires one graph construction and $2k - 1$ computations of the shortest path, which are executed in

[13] This does not contradict the optimality of MHM, because it is optimal regarding the order of records in the path, while we rotate the records in the path to get more utility.
[14] It is assumed that $n = |T| = |P| \geq k$.
[15] Please remember that the maximum size of a group is limited to $2k - 1$, i.e., $|G_p| < 2k$.

$O(nk+nk^2)$. Finding an improvement based on 1-opt or 2-opt heuristics requires considering all edges in H in the worst case and consumes $O(n^2)$ time. The next step is to sort the improvements found and applying some of them in a greedy manner. The complexity of this part can be calculated based on the predefined pool size p_n, which may be set in a way that the complexity of the sort algorithm $O(p_n \log p_n)$ is dominated by the complexity of other parts of the algorithm. Next part of the algorithm is about to calculate *Score*, and *Rank* for constructing H'. [16] Calculating the *Score* and *Rank* of the groups requires $O(n^2)$ and $O(c+c\log c)$, respectively. However, running the shortest path problem in H' and finding a negative cycle requires $O(n+c+n^2)$ operations. Finally, applying the improvements (if any), needs at most $O(c)$ changes in the assignment, which is dominated by $O(n^2)$ complexity of the previous lines. Therefore, the time complexity of our algorithm is bounded by $O(n^2)$. However, as our experiments confirm, the time complexity is not an issue in practice. [17]

5 Experimental Results

In this section, we present the evaluation results of our algorithm.

5.1 Configuration Settings

Three standard benchmark data sets in the SDC are **Tarragona** , **Census** , and **EIA** [20], which are introduced in Table 1. These data sets are used in multiple papers such as [9, 10, 13]. We have also introduced a **Synthetic** data set of 10000 records with 10 attributes. The data set contains 10 clusters of normally distributed data point around random cluster centers. Table 2 shows the results of ICSM on these data sets. In this table, Iteration denotes the number of rounds with improvements in the assignment. Moreover, the $MAX\_ITERATIONS$ for **Synthetic** data set is set to 20 and 400, but is infinity for other data sets.

Additionally, the average results on 25 synthetic uniform distributed data sets are reported in Table 2. Each data set consists of $n=100$ data records with $d=10$ numerical attributes drawn from $[-1000, 1000]$ interval by simple random sampling. Another simulated test set contains 25 synthetic data sets, somehow similar to [10] [18] and [19]. Each data set has $c=5$ clusters, each of them contains $n=20$ multivariate Gaussian distributed data records around. All data records of a cluster center have 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 $[0.1, 0.3]$. 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 two 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 the data sets more clustered.

The information loss of ICSM for these data sets is reported in Table 2. The average values of information utility improvements over MDAV method in terms of IL are also denoted as *Gain*. Moreover, the run

Table 1. Benchmark data sets for microaggregation comparison [20]

data set	number of data records (n)	number of numeric attributes (d)
Tarragona	834	13
Census	1080	13
EIA	4092	11
Synthetic	10000	10

Table 2. Evaluation results of the proposed algorithm, ICSM in comparison with MDAV. $Gain = 1-(IL_{ICSM})/(IL_{MDAV})(\%)$ is used.

Data set	k	IL_{MDAV}	IL_{ICSM}	Iterations	*Gain*	$Time_{MDAV}$ (s)	$Time_{ICSM}$ (s)
Tarragona	3	16.93	14.81	174	12.5	<1	1
	5	22.46	20.69	238	7.9	<1	1
	10	33.19	30.7	127	7.5	<1	1
Census	3	5.69	4.85	203	14.8	<1	2
	5	9.09	7.78	205	14.4	<1	2
	10	14.16	11.93	210	15.7	<1	3
EIA	3	0.48	0.36	29	25	<1	6
	5	1.67	0.78	39	53.3	<1	8
	10	3.84	2.24	142	41.7	<1	26
Synthetic	3	6.74	6.55	20	2.8	1	15
			6.17	400	8.5		338
	5	10.3	10.14	20	1.6	1	13
			9.02	400	12.4		332
	10	14.94	14.78	20	1.1	1	12
			12.72	400	14.9		300
Sim-1	3	32.68	27.95	-	14.5	<1	<1
	5	48.65	42.87	-	11.9	<1	<1
	10	67.52	63.92	-	5.3	<1	<1
Sim-2	3	14.32	12.44	-	13.1	<1	<1
	5	23.56	20.13	-	14.6	<1	<1
	10	37.18	31.91	-	14.2	<1	<1
Sim-3	3	15.51	12.65	-	18.5	<1	<1
	5	23.96	20.31	-	15.3	<1	<1
	10	37.73	32.61	-	13.6	<1	<1

[16] However, there is no need to construct it in practice, since the shortest path algorithm over H' may be conducted using a simple index redirection, i.e., we resort the indexes of elements in the adjacency matrix of H, so that to include only finite edges of H'.

[17] The ICSM is an anytime algorithm that can be stopped before its completion.

[18] In the original paper, the number of clusters (c) and data records in a cluster (G_i) was chosen randomly.

time of the algorithm (best time out of 10 executions, excluding the time required for reading the input file), is also presented. In all tests, a regular PC with a Core i7, 2.5 GHz CPU and Windows 7 64-bits is used. The pool size p_n is set to 100.

The results confirm that ICSM always improves information loss in comparison with MDAV. This improvement reaches up to 53% for a clustered data set such as **EIA** . These improvements are at the expense of running time of the ICSM. However, this is not a major drawback in practice, since the whole process of anonymization is an off-line task.

5.2 Comparison of the ICSM with other Microaggregation Algorithms

To our knowledge, most results reported here are among the best results in the literature in terms of IL. This is due to the effectiveness of the introduced constraints toward optimizing IL of a solution. To show the superiority of ICSM, we have also compared the results of ICSM with nine other microaggregation methods on **Tarragona** and **Census** benchmark data sets. [19]

Table 3 shows the complexities of these methods. The evaluation results are also presented in Figure 5. [20] In all experiments, ICSM achieves the best results in terms of IL. These superiority is the results of considering the two main constraints of an optimal solution of a microaggregation algorithm, i.e., the exchange and migration constraints.

Table 3. Some characteristics of microaggregation methods used in comparison

Method	Classification	Complexity	Reference
MDAV	Fixed size	$O(n^2/k)$	[7]
GSMS-T2	Data Oriented	$O(n^2 \log n)$	[10]
μ-Approx	Data Oriented	$O(n^2)$	[3]
M-d	Data Oriented	$O(n^2)$	[8]
NPN-MHM	Data Oriented	$O(n^2)$	[8]
MDAV-MHM	Data Oriented	$O(n^2/k)$	[8]
MD-MHM	Data Oriented	$O(n^3/k)$	[8]
TFRP	Data Oriented	$O(n^2/k)$	[9]
FDM	Data Oriented	$O(n^2 \log n)$	[15]

[19] The results on **EIA** and **Synthetic** data sets show similar improvements and are omitted for brevity.
[20] All information loss values in Figure 5 are based on the referenced papers in Table 3.

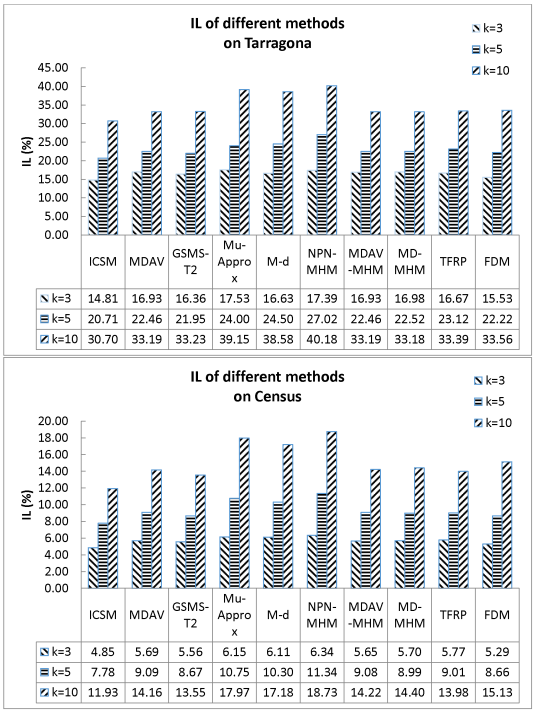

Figure 5. Comparison of ICSM with eight other microaggregation methods on **Tarragona** (top) and **Census** (bottom) data sets.

5.3 Random Initialization of the ICSM

In this section, we repeat the experiments of the previous section, but a random assignment of records to clusters is utilized for initialization. These experiments are repeated for 1600 different perturbation of MDAV assignments for each data set and k, where the assignments of at most 20 randomly selected records are changed. We compared the IL results of the ICSM with LS [14]. The results are shown in Table 4. This table confirms the superiority of the ICSM, since it usually reaches a more useful protected data set for the specified value of k, while at the same time required less random restarts than LS.

6 Conclusion

In this paper, we presented a novel approach to reduce the information loss of microaggregation. The method involves 1-opt, 2-opt, and n-opt heuristics to satisfy migration and exchange constraints. Based on the proposed notation, these heuristics may be represented as negative cycles in a graph. Applying these improvements in an iterative manner may reduce the IL, especially in the first iterations of the algorithm. Regarding the evaluation results on multiple standard and synthetic data sets with different distributions, it has been shown that the method is superior to other heuristics proposed in the literature. This is due to the generality and effectiveness of the heuristics to satisfy some necessary conditions of an optimal solution. It is

Table 4. Comparison of the best IL of ICSM and LS [14] methods.

Data set	k	IL_{ICSM}	IL_{LS}
Tarragona	3	14.54	14.68
	4	17.18	17.23
	5	20.25	20.3
	10	30.34	30.23
Census	3	4.75	4.85
	4	6.21	6.66
	5	7.5	7.86
	10	11.74	11.94
EIA	3	0.35	0.45
	4	0.49	0.58
	5	0.74	1.2
	10	1.95	2.45

interesting to consider other heuristics to improve information utility after microaggregation, particularly for n-opt heuristic. This is considered as our future work to achieve more useful protected data.

Acknowledgment

This work has been supported by ITRC (Iran Telecommunication Research Center) under contract No. 12200/500.

References

[1] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5): 557–570, 2002. ISSN 0218-4885.

[2] 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.

[3] Josep Domingo-Ferrer, Francesc Sebé, and Agusti Solanas. A polynomial-time approximation to optimal multivariate microaggregation. *Computers & Mathematics with Applications*, 55(4):714–732, 2008.

[4] J. Domingo-Ferrer and J.M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002. ISSN 1041-4347.

[5] 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. ISSN 1041-4347.

[6] A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Comission for Europe*, 18(4):345–354, 2001. ISSN 0167-8000.

[7] Josep Domingo-Ferrer and Vicenç Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212, 2005. ISSN 1384-5810.

[8] 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. ISSN 1066-8888.

[9] Chin-Chen Chang, Yu-Chiang Li, and Wen-Hung Huang. TFRP: An efficient microaggregation algorithm for statistical disclosure control. *Journal of Systems and Software*, 80(11):1866 – 1878, 2007. ISSN 0164-1212.

[10] Costas Panagiotakis and Georgios Tziritas. Successive group selection for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1191–1195, 2013. ISSN 1041-4347.

[11] M. Laszlo and S. Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005. ISSN 1041-4347.

[12] Marc Solé, Victor Muntés-Mulero, and Jordi Nin. Efficient microaggregation techniques for large numerical data volumes. *International Journal of Information Security*, 11(4):253–267, 2012. ISSN 1615-5262.

[13] Jun-Lin Lin, Tsung-Hsien Wen, Jui-Chien Hsieh, and Pei-Chann Chang. Density-based microaggregation for statistical disclosure control. *Expert Systems with Applications*, 37(4):3256 – 3263, 2010. ISSN 0957-4174.

[14] Michael Laszlo and Sumitra Mukherjee. Iterated local search for microaggregation. *Journal of Systems and Software*, 100:15–26, 2015.

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

[16] Sergio Martínez, David Sánchez, and Aida Valls. Semantic adaptive microaggregation of categorical microdata. *computers & security*, 31(5):653–672, 2012.

[17] Jun-Lin Lin, Pei-Chann Chang, Julie Yu-Chih Liu, and Tsung-Hsien Wen. Comparison of microaggregation approaches on anonymized data quality. *Expert Systems with Applications*, 37(12): 8161–8165, 2010.

[18] David Rebollo-Monedero, Jordi Forné, and Miguel Soriano. An algorithm for k-anonymous microaggregation and clustering inspired by the design of distortion-optimized quantizers. *Data & Knowledge Engineering*, 70(10):892–921, 2011. ISSN 0169023X.

[19] Reza Mortazavi, Saeed Jalili, and Hojjat Gohargazi. Multivariate microaggregation by iterative optimization. *Applied Intelligence*, 39(3): 529–544, 2013. ISSN 0924-669X.

[20] Ruth Brand. Microdata protection through noise addition. In Josep Domingo-Ferrer, editor, *Inference Control in Statistical Databases*, volume 2316 of *Lecture Notes in Computer Science*, pages 97–116. Springer, 2002. ISBN 978-3-540-43614-0.

7 Appendix

Symbols used in this paper are as follow:

Table 5. Description of symbols used in the paper

Symbol	Description
T	the original numerical data set (in standard normal form for $d \geq 1$)
k	the minimum number of records in each group after microaggregation
SSE	sum of within-group Squared Error
n	number of records in T
IL	information Loss
SST	total sum of Squared Error (for the whole data set)
d	number of attributes of T
x_i	the i-th numerical record in T
G_p	the p-th group of records
c	number of groups
x_{pj}	the j-th record of G_p
$\bar{x}_p$	the centroid of G_p
$PSSE$ $(x_{pi} \rightarrow G_p)$	partial summand of SSE
S_{opt}	the optimal assignment of records to clusters in a microaggregation problem
SP	shortest path
p_n	pool size to store potential improvements

Reza Mortazavi received his Ph.D. degree in computer engineering from Tarbiat Modares University (TMU) in 2015. He received the M.S. degree in Software Engineering from K.N.Toosi University of Technology in 2007, and the B.S. 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.S. degree in computer science from Sharif University of Technology in 1985. Since 1992, he has been an 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.