International Journal of Engineering

Journal Homepage: www.ije.ir

Repeated Record Ordering for Constrained Size Clustering

R. Mortazavi*

School of Engineering, Damghan University, Damghan, Iran

PAPER INFO

Paper history:
Received 15 March 2020
Received in revised form 09 April 2020
Accepted 17 June 2020

Keywords:
Constrained Clustering
Microaggregation
Data Privacy

ABSTRACT

One of the main techniques used in data mining is data clustering, which has many applications in computer science, biology and social sciences. Constrained clustering is a type of clustering in which side information provided by the user is incorporated into current clustering algorithms. One of the well researched constrained clustering algorithms is called microaggregation. In a microaggregation technique, the algorithm divides the dataset into groups containing at least k members, where k is a user-defined parameter. The main application of microaggregation is in Statistical Disclosure Control (SDC) for privacy preserving data publishing. A microaggregation algorithm is qualified based on the sum of within-group squared error, SSE. Unfortunately, it has been proven that the optimal microaggregation problem is NP-Hard in general, but the special univariate case can be solved optimally in polynomial time. Many heuristics exist for the general case of the problem that are founded on the univariate case. These techniques order multivariate records in a sequence. This paper proposes a novel method for record ordering. Starting from a conventional clustering algorithm, the proposed method repeatedly puts multivariate records into a sequence and then clusters them again. The process is repeated until no improvement is achieved. Extensive experiments have been conducted in this research to confirm the effectiveness of the proposed method for different parameters and datasets.

***doi**: 10.5829/ije.2020.33.07a.13*

1. INTRODUCTION

Nowadays, there is a considerable demand for real-world datasets in various data mining tasks. However, the privacy of involved entities usually agitates data owners about the usage of such information [1, 2]. Privacy preserving data publishing is the task that addresses the problem. The problem is also investigated in research communities of the Internet of Things (IoT) [3, 4] and Statistical Disclosure Control (SDC). Usually, the privacy requirement in terms of Disclosure Risk (DR) is formalized using a computational privacy model, which can then be realized by an implementation method. The main idea of different solutions is based on changing the original data records to preserve the privacy of involved entities. Such changes decrease the utility of published data which is stated by Information Loss (IL). It is desired to minimize both the competing indices of DR and IL, which is a challenging multi-objective optimization task [5].

One of the most famous computational privacy models is called k-anonymity [6]. In a k-anonymous dataset, for each set of identifying attributes, there exist at least k records. Therefore, an intruder who knows some attributes of an entity cannot limit its record data to a small group, i.e., a group with less than k members. Microaggregation is a perturbative approach to realize k-anonymity. It was initially developed for numerical data volumes, while it can also be used for other types of datasets [7]. A microaggregation technique tries to cluster the dataset records into groups with at least k members and then aggregates them into their centroids. The centroids are then substituted for the original records and published for public usage. In other words, the original entries are masked using their associated centroids. The replacement decreases the details of the published values, which results in IL. For microaggregation algorithms, IL is usually quantified in terms of the sum of within-group

*Corresponding Author Institutional Email: *r_mortazavi@du.ac.ir* (R. Mortazavi)

squared error (SSE).

Unfortunately, it has been proven that given the privacy parameter k, the optimal microaggregation problem is NP-hard in general [8]. Still, the univariate instance can be optimally solved in polynomial time using the Mukherjee and Hansen Microaggregation (MHM) algorithm [9]. Some heuristic approaches try to map the general multivariate microaggregation problem to the univariate case [10, 11]. For example, the NPN-MHM algorithm [10] traverses all records in a Nearest Point Next fashion starting from the farthest record from the dataset centroid to put them in a sequence and then applies MHM on the output ordering. Similarly, MDAV-MHM [10] clusters the dataset using a traditional microaggregation algorithm, Maximum Distance to Average Vector (MDAV) [7] and then visits all records, group by group. Mortazavi et al. proposed Improved MHM (IMHM[‡]) [11], which accelerates MHM and uses it in multivariate microaggregation. However, existing techniques are not general and usually suffer from increased IL when the dataset has an internal structure and is naturally clustered. For instance, NPN-MHM is more useful in anonymizing datasets with very separated clusters, but for clustered data with moderate gaps or skewed data, the CBFS–MHM and MDAV–MHM produce the best results [10]. Similarly, IMHM [11] is more successful when k is small and the dataset is clustered, but for homogeneous datasets, it produces more useful anonymized versions when k is large. Additionally, comparing the results of some recent heuristics with proved lower bounds of the problem [12] shows large gaps in some cases.

The primary contribution of this paper is to propose an innovative ordering technique in which multivariate data records are ordered in a sequence while considering the internal structure of the dataset using a conventional clustering method. Additionally, it is shown that the process of converting the output of a clustering algorithm to a sequence can be repeated that in turn results in considerable improved IL. Extensive experiments in this research show the advantage of the proposed method in terms of data utility in comparison with similar previous techniques.

The remainder of the paper is structured as follows. Section 2 formalizes the microaggregation problem. Section 3 reviews some related microaggregation algorithms. Section 4 describes the proposed method. Experimental results are reported in Section 5. Finally, Section 6 concludes the paper.

2. MICROAGGREGATION PROBLEM

In this section, the problem of microaggregation is formalized. Assume a dataset T of n numerical records in a d-dimensional space, i.e., $T = \{x_1, x_2, \dots, x_n\}$ where $x_i \in \mathbb{R}^d$. Given an input value k as the privacy parameter, the microaggregation algorithm aims to partition the whole dataset T into c non-overlapping groups $G_1, \dots, G_c$ each with at least k members. The objective of microaggregation techniques as an optimization problem is to minimize the SSE, which aims to obtain clusters of similar records. This measure is shown in Equation (1).

$$SSE = \sum_{p=1}^{c} \sum_{j=1}^{|G_p|} \left(x_{pj} - \overline{x_p}\right)^T \left(x_{pj} - \overline{x_p}\right) \qquad (1)$$

In Equation (1), x_{pj} is record j of group G_p, and $\overline{x}_p$ denotes the centroid of G_p, i.e., $\overline{x}_p = \sum_{j=1}^{|G_p|} x_{pj} / |G_p|$. The value is usually divided by the Sum of Squares Total (SST) to normalize IL. SST is related to the dataset itself and is invariant to the microaggregation algorithm or the privacy model parameters. It is formulated in Equation (2).

$$SST = \sum_{i=1}^{n} (x_i - \bar{x})^T (x_i - \bar{x}) \qquad (2)$$

In Equation (2), $\bar{x}$ is the centroid of the whole dataset, i.e., $\bar{x} = \sum_{i=1}^{n} x_i / n$. The normalized measure $IL = SSE/SST * 100\%$ is always between 0 and 100%, where lower values of IL indicate less utility degradation due to microaggregation.

3. RELATED WORKS

It was shown by Domingo-Ferrer and Mateo-Sanz that in an optimal constrained size clustering, each group contains at most $2k - 1$ records [13]. A polynomial-time technique was developed by Hansen and Mukherjee for univariate microaggregation that is called MHM [9]. The MHM first sorts univariate records and then creates a directed acyclic graph in which each arc in the graph matches a valid group that may be a cluster in the optimal solution. The authors showed that the optimal univariate microaggregation problem is reduced to computing the shortest path in the graph. A cluster exists in the optimal partition if its equivalent arc is in the computed shortest path. The complexity of the technique is $O(\max(n \log n, k^2 n))$. Mortazavi *et al.* introduced an improved implementation of the MHM called IMHM [11] that makes use of incremental weight computation of graph arcs to improve the complexity of graph construction to $O(kn)$ operations. The authors generalized the application of IMHM for multivariate datasets in an iterative optimization process, but the experiments show that the user has to carry out different experiments with multiple parameters, which is a time-consuming task.

[‡] The pseudo-code of the IMHM is described briefly in the Appendix.

The optimal property of MHM provides a hopeful tactic to solve the challenging problem of the multivariate microaggregation. However, sorting multivariate records for optimal microaggregation is not well-defined. Therefore, different heuristics are devised in literature to sequence multivariate records. Domingo-Ferrer *et al.* [10] proposed some heuristics, such as the Nearest Point Next MHM (NPN-MHM), MDAV-MHM, and Centroid-Based Fixed-Size MHM (CBFS-MHM) to order records and form a sequence of them. Then, MHM is applied to records on the path. However, their reports show that their approach is usually far from optimal, especially for clustered datasets. Monedero *et al.* used two projection methods, i.e., Principal Component Analysis (PCA) and Z-score, to reduce the dimension of the underlying dataset to one [14]. In the PCA technique, the first principal component of the dataset is utilized to sort data records. The Z-score algorithm orders multivariate records based on the sum of their Z-scores. Again, there is a significant distance to optimal solutions in both methods. Soria-Comas and Domingo-Ferrer presented a method to satisfy the differential privacy requirement [15] through univariate microaggregation [9]. Additionally, Mortazavi and Jalili introduced the Fast Data-oriented Microaggregation algorithm (FDM) [16] that produces an optimal assignment of records with respect to their Travelling Salesman Problem (TSP[§]) tour for a continuous range of the privacy parameter k. However, the running time to compute the TSP tour of multivariate records is considerable. More recently, Khomnotai *et al.* devised the Iterative Group Decomposition (IGD) technique [17] to refine the solution of a microaggregation algorithm by either shrinking or decomposing its clusters. Unfortunately, none of the mentioned methods can achieve near-optimal solutions. They are usually useful for particular datasets with pre-specified data distribution or very limited ranges of k. Moreover, the methods in the literature are somehow hard-coded with complex parameters that limit their flexibility in practice. It is therefore desired to devise a general method that can produce more useful anonymized datasets, which is addressed in the next section.

4. PROPOSED MICROAGGREGATION ALGORITHM

In this section, the Repeated record Ordering heuristic for multivariate Microaggregation, RepOrdMic is detailed. Briefly, the algorithm accepts an initial clustering of records and traverses all records group-by-group to complete a sequence (ordering) of all records. In each group, all records are visited using a TSP heuristic, and then the nearest unexplored group is processed. After all records were added to the sequence, the IMHM is utilized to produce a (constrained) clustering. The process is repeated until no significant improvement is achieved. Algorithm 1 shows the pseudo-code of the proposed method[**]. The algorithm accepts the normalized dataset T, the privacy parameter k, and an initial clustering label lbl_{in} as inputs, and produces the perturbation error SSE and labels of assigned records to constrained size groups lbl_{out} as outputs. The function initially creates an empty sequence Seq to store the total ordering of multivariate records in Step 1. Step 2 finds the farthest record x_f from the whole dataset centroid and then stores it in the current record x_c in Step 3. In Steps 4 to 9, all records in T are visited group-by-group and the order of visiting them is saved in Seq. In Step 4, the group label of x_c is considered as the current group, G_c. If the current group has only one member, the algorithm continues to process other groups (Step 6-1). Otherwise, the algorithm looks for the most distant point from the current record x_c among current group members and adds it to the end of Seq. Other records in the current group will be inserted between these two group members. The process utilizes an idea inspired by the nearest insertion heuristic to solve TSP for entering all records in the current group to the Seq. Steps 7 to 9 repeatedly choose an unseen record with minimum distance to its nearest neighbor among the current group members in Seq. Then, they insert it between the two consecutive records (Figure 1) for which such an insertion causes the minimum increase in the total sequence length of Seq. In other words, the insertion has to minimize $\text{len}(Seq) = \sum_{i=1}^{|Seq|-1} D(Seq[i+1], Seq[i])$ where $D(.)$ denotes the Euclidean distance operator. For example, by inserting x_t between two records x_i and x_{i+1}, $D(x_i, x_t)$ and $D(x_t, x_{i+1})$ are added to the total length of Seq, but the distance between x_i and x_{i+1}, i.e., $D(x_i, x_{i+1})$ is subtracted from the total length. These cost values are computed in Step 9, and the minimum one is added to Seq in Step 10. The process continues until all records in the current group are added to Seq. In Step 12, if no unseen record remains, the algorithm goes to Step 14, otherwise the nearest unseen record in the whole dataset to the last record in Seq is chosen as the current record in Step 12-2, and the process of record ordering continues from Step 3. After ordering all records, the algorithm applies the optimal univariate microaggregation algorithm IMHM on it to produce a clustering that satisfies the size constraint and computes its SSE in Step 13. This clustering can be fed again to the algorithm to reorder all records and improve SSE (Step 15). If SSE is not decreased significantly, the function terminates, and the last computed SSE and the final

[§] Please recall that given a list of points, the TSP is to find the shortest possible route that visits each point and returns to the origin [16].

[**] An illustrative example of the algorithm execution on a small dataset is provided in the supplementary document of the paper.

clustering labels are returned as the algorithm outputs in Step 16.

4. 1. Analysis of the Algorithm The idea of using an initial clustering makes it possible to capture the inherent structure of the underlying dataset. Moreover, processing all records in a group-by-group fashion enables the process to focus on records of the current group rather than the whole dataset that makes the algorithm more efficient. The proposed method consists of repeated steps of record ordering in a sequence and applying IMHM. The SSE of each iteration of the loop is not worse than its value in the previous iteration since the ordering procedure meets records in a group-by-group manner and IMHM does not change the order of records in each group. Therefore, the SSE of each iteration improves gradually, or no significant decrement occurs at the last step, and the algorithm stops. It is also notable that the algorithm stops necessarily since the lower-bounded SSE cannot improve infinitely[††].

The space complexity of the proposed method for $|T| = n$ is $O(n)$ that is used for storing the ordering of records in Seq and computing the optimal clustering in IMHM. However, the runtime complexity is

Algorithm 1. The pseudo-code of the RepOrdMic

```
Input: $T = \{x_1, x_2, \dots, x_n\}$: original dataset, $k$: the privacy parameter, $lbl_{in}$: initial clustering labels
Output: $SSE$: microaggregation error, $lbl_{out}$: the label of records to groups with at least $k$ members
1       Initialize the sequence $Seq$ to NULL
2       Find the farthest record $x_f$ from the dataset centroid
3       $x_c \leftarrow x_f$
4       Set the group label of $x_c$ as the current group $G_c$, i.e., $G_c \leftarrow lbl_{in}[x_c]$
5       Add $x_c$ to the end of $Seq$
6       If the current group size is less than 2
6-1         Goto Step 12.
        Else
6-2         Find the farthest record from $x_c$ in the current group and add it to the end of $Seq$.
7       Foreach consecutive records of $G_c$ in $Seq$, $x_i$, and $x_{i+1}$
8           Foreach record $x_t$ in the $G_c$ that is not in $Seq$
9               Compute the cost of $x_t$ addition to $Seq$ at position $i$, i.e., $cost_{t,i} \leftarrow D(x_i, x_t) + D(x_t, x_{i+1}) - D(x_i, x_{i+1})$.
10      Find the minimum cost, say $cost_{t^*,i^*}$ and insert $x_{t^*}$ between $x_{i^*}$ and $x_{i^*+1}$.
11      If there exists any record of the current group that is not in $Seq$, Goto Step 7.
12      If there is not any unseen group
12-1        Goto Step 14
        Else
12-2        From any unseen groups, find the nearest record to the last record in $Seq$, set it as the current record $x_c$, and Goto Step 3.
13      Apply IMHM to $Seq$, to compute information loss and new labels, and save them in $SSE$ and $lbl_{out}$, respectively.
14      If $SSE$ is improved significantly
15          $lbl_{in} \leftarrow lbl_{out}$
            Goto Step 2
        Else
16          Return the last $SSE$ and $lbl_{out}$
```

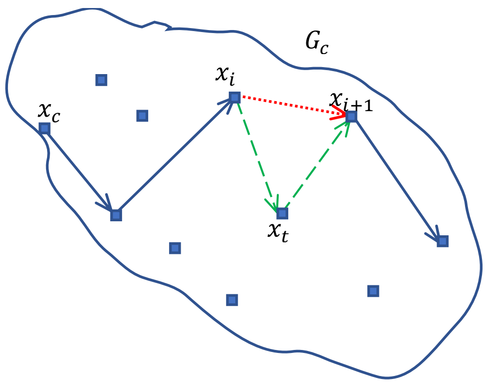

Figure 1. Adding $\boldsymbol{x_t}$ between $\boldsymbol{x_i}$ and $\boldsymbol{x_{i+1}}$ in the current group $\boldsymbol{G_c}$. Dashed green and dotted red lines indicate inclusion and removal, respectively

considerable. It is assumed that an initial clustering is provided before the first iteration. This clustering can be used for multiple values of the privacy parameter k and has to be computed once so that its execution time can be safely discarded. Finding the farthest record from the dataset centroid in Step 2 requires $O(n)$ operations. In the following steps of the algorithm, each iteration orders records of each group in a sequence. Except for the first step that processes the initial clustering labels, the following clustering results have $O(k)$ records in each group since they are the output of IMHM. Therefore, for each of $O(k)$ records in a clustering with $O(n/k)$ groups, the cost values can be computed in $O(k)$ operations. The process has to be repeated $O(k)$ times in each group to cover all records in the group, thus $O(k^3)$ operations are

[††] Note that the number of different orderings is limited and the algorithm visits each ordering at most once, so it has to stop. In practice, the iterations are broken when no significant improvement in SSE is achieved.

needed for each group. Finding the nearest unseen record of other groups is accomplished in $O(n)$ and is repeated $O(n/k)$ times. Hence, the ordering is completed in $O\left(\frac{n}{k}.k^3+n.\frac{n}{k}\right)=O\left(n.k^2+\frac{n^2}{k}\right)$ computations. The univariate microaggregation algorithm can be implemented efficiently in $O(nk)$ operations. As a result, for l iterations, the algorithm complexity is $O\big(l.n.(k^2+k+n/k)\big)$. However, it is notable that the whole process is an offline task, and the runtime is usually not a bottleneck, but the quality of the produced clustering in terms of SSE is more important.

As a side note, for an efficient implementation of the ordering process, it can be seen that after inserting a record to sequence Seq, most of the previously computed cost values remain the same and can be reused, but a small number of them has to be computed or updated.

5. EXPERIMENTAL RESULTS

A prototype of the proposed method was implemented in Microsoft Visual C++ 2019 in release mode. All evaluations are conducted within Windows 10 operating system on a regular laptop with Intel Core i5-8265U 1.60 GHz CPU and 8 GB of main memory. For initial clustering, different outputs of the *k*-means clustering algorithm are used for a number of clusters between 1 annd 200. Additionally, the iterations are broken when $\Delta SSE<1e-7$.

Experiments were performed on three real-world benchmark datasets that are usually used for the evaluation of microaggregation algorithms. Benchmark datasets that are used in related previous studies [16, 17] are described in Table 1. All datasets contain numeric attributes without any missing values.

Table 2 shows information loss for various values of k. The proposed algorithm, namely RepOrdMic, is compared with MDAV [11], MDAV-MHM [10], IMHM [11], and IGD [17] methods. The results show that IL increases when k becomes greater for all microaggregation algorithms. For instance, $IL_{MDAV-MHM}$ for Census and $k=3$ is 5.65, which is increased to 14.22 for $k=10$. Additionally, IL of Tarragona dataset is generally higher than the other two datasets since it is known as a sparse dataset, which increases the cost of the anonymization process. The classic methods MDAV [11] and MDAV-MHM [10] are reported as reference techniques but do not produce any competing results in total. Similarly, the results of IGD are always worse than the winner methods. The IMHM [11] is more successful in the anonymization of Tarragona and Census for $\boldsymbol{k}=\mathbf{10}$, while the difference between IMHM and RepOrdMic is negligible in these cases. RepOrdMic, the proposed method, achieves the best results in other cases. For example, for EIA and k=10, RepOrdMic has improved the outputs of MDAV, MDAV-MHM, IMHM, and IGD by 48.18%, 47.21%, 8.29%, and 1.48%, respectively. In brief, the results indicate that RepOrdMic is successful in producing more useful datasets in 7 out of 9 experiment sets.

All elapsed times of the proposed method, excluding initial clustering, read and write disk operations, and (de)normalization are shown in Table 3. These values are reported for 200 different number of initial clusters from 1 to 200 that are produced by *k*-means seeded with 0. The runtime of the algorithm for EIA is much larger than the other two cases since EIA is a large clustered numeric volume that makes it difficult and time-consuming for the algorithm to satisfy the privacy requirement. The experiments also show a decreasing runtime trend when $\boldsymbol{k}$ increases since for small values of $\boldsymbol{k}$, the runtime of Step 12-2 in Algorithm 1 dominates the runtime of other parts, but the behavior changes when $\boldsymbol{k}$ becomes larger. In brief, given an initial clustering, the algorithm is efficient and general; it usually terminates in a reasonable time regardless of the privacy parameter and underlying distribution or structure of the dataset.

Notably, the experiments can be extended to evaluate other important issues about the proposed method such as its effect on the proximity-based attack [18], its application for anonymization of complex structures such as graphs [19], and the impact of using other initial clustering techniques such as x-means [20] or consensus clustering [21].

TABLE 1. Standard benchmark datasets for microaggregation comparison [16]

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

TABLE 2. Information loss comparison for various standard datasets. Best $\boldsymbol{IL}$ values are bolded

Dataset	$\boldsymbol{k}$	**MDAV**	**MDAV-MHM**	**IMHM**	**IGD**	**RepOrdMic**
Tarragona	3	16.93	16.93	16.93	15.60	14.80
	5	22.46	22.46	22.18	21.31	21.13
	10	33.19	33.19	30.78	32.87	31.13
Census	3	5.69	5.65	5.37	5.33	5.01
	5	9.09	9.09	8.42	8.37	7.94
	10	14.16	14.22	12.23	12.65	12.74
EIA	3	0.48	0.41	0.374	0.39	0.369
	5	1.67	1.26	0.76	0.76	0.75
	10	3.84	3.77	2.17	2.02	1.99

TABLE 3. The runtime of RepOrdMic for 200 initial clustering labels.

Dataset	*k*	Total time (sec)	Total Iterations	Avg Iterations per clustering	Time per iteration (msec)
Tarragona	3	5.4	2525	12.63	2.14
	5	4.6	2755	13.78	1.67
	10	5	2988	14.94	1.67
Census	3	7.3	2368	11.84	3.08
	5	6.6	2504	12.52	2.64
	10	5.7	2527	12.64	2.26
EIA	3	51.3	1643	8.22	31.22
	5	36.8	1811	9.06	20.32
	10	25.9	1922	9.61	13.48

6. CONCLUSIONS

This paper presents a novel microaggregation algorithm based on the repeated ordering of multivariate records and mapping them to optimal univariate microaggregation algorithm IMHM. The process of ordering and applying IMHM is repeated until no significant improvement is achieved in terms of SSE. The output quality of the proposed method is usually better than similar methods in terms of IL. Extensive experiments on real-world datasets for different values of the privacy parameter k confirm that the algorithm is an efficient and general approach for practical usages. A promising extension of the proposed technique for future study is the way the records are ordered in a sequence.

7. REFERENCES

1. Erfani, S.H. and Mortazavi, R., "A Novel Graph-modification Technique for User Privacy-preserving on Social Networks", ***Journal of Telecommunications and Information Technology***, Vol. 3, (2019), 27-38. DOI: 10.26636/jtit.2019.134319
2. Mortazavi, R. and Erfani, S.H., "An effective method for utility preserving social network graph anonymization based on mathematical modeling", ***International Journal of Engineering, Transactions A: Basics*** Vol. 31, No. 10, (2018), 1624-1632. DOI: 10.5829/ije.2018.31.10a.03
3. Bypour, H., Farhadi, M. and Mortazavi R., "An Efficient Secret Sharing-based Storage System for Cloud-based Internet of Things", ***International Journal of Engineering***, ***Transactions B: Applications***, Vol. 32, No. 8, (2019), 1117-1125. DOI: 10.5829/ije.2019.32.08b.07
4. Gheisari, M., Wang, G., Bhuiyan, M.Z.A. and Zhang W., "Mapp: A modular arithmetic algorithm for privacy preserving in IoT", In IEEE International Symposium on Parallel and Distributed Processing with Applications and IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), (2017), 897-903. DOI: 10.1109/ISPA/IUCC.2017.00137
5. Mortazavi, R. and Jalili, S., "Preference-based anonymization of numerical datasets by multi-objective microaggregation", ***Information Fusion***, Vol. 25, (2015), 85-104. https://doi.org/10.1016/j.inffus.2014.10.003
6. Sweeney, L., "k-anonymity: A model for protecting privacy", ***International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems***, Vol. 10, No. 5, (2002), 557-570. https://doi.org/10.1142/S0218488502001648
7. Domingo-Ferrer, J. and Torra, V., "Ordinal, continuous and heterogeneous k-anonymity through microaggregation", ***Data Mining and Knowledge Discovery***, Vol. 11, No. 2, (2005), 195-212. https://doi.org/10.1007/s10618-005-0007-5
8. Oganian, A. and Domingo-Ferrer, J., "On the complexity of optimal microaggregation for statistical disclosure control", Statistical Journal of the United Nations Economic Commission for Europe, Vol. 18, No. 4, (2001), 345-353.
9. Hansen, S.L. and Mukherjee, S., "A polynomial algorithm for optimal univariate microaggregation", ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 15, No. 4, (2003), 1043-1044. DOI: 10.1109/TKDE.2003.1209020
10. Domingo-Ferrer, J., Martínez-Ballesté, A., Mateo-Sanz, J.M. and Sebé, F., "Efficient multivariate data-oriented microaggregation", ***The VLDB Journal***, Vol. 15, No. 4, (2006), 355-69. https://doi.org/10.1007/s00778-006-0007-0
11. Mortazavi, R., Jalili, S. and Gohargazi, H., "Multivariate microaggregation by iterative optimization", ***Applied Intelligence***, Vol. 39, No. 3, (2013), 529-544. https://doi.org/10.1007/s10489-013-0431-y
12. Aloise, D., Hansen, P., Rocha, C. and Santi, É., "Column generation bounds for numerical microaggregation", ***Journal of Global Optimization***, Vol. 60, No. 2, (2014), 165-182. https://doi.org/10.1007/s10898-014-0149-3
13. Domingo-Ferrer, J. and Mateo-Sanz, J.M., "Practical data-oriented microaggregation for statistical disclosure control", ***IEEE Transactions on Knowledge and Data Engineering***, Vol. 14, No. 1, (2002), 189-201.
14. Monedero, D.R., Mezher, A.M., Colomé, X.C., Forné, J. and Soriano, M., "Efficient k-anonymous microaggregation of multivariate numerical data via principal component analysis", ***Information Sciences***, Vol. 503, (2019), 417-443. https://doi.org/10.1016/j.ins.2019.07.042
15. Soria-Comas, J. and Domingo-Ferrer, J., "Differentially private data publishing via optimal univariate microaggregation and record perturbation", ***Knowledge-Based Systems***, Vol. 153, (2018), 78-90. https://doi.org/10.1016/j.knosys.2018.04.027
16. Mortazavi, R. and Jalili, S., "Fast data-oriented microaggregation algorithm for large numerical datasets", ***Knowledge-Based Systems***, Vol. 67, (2014), 195-205. https://doi.org/10.1016/j.knosys.2014.05.011
17. Khomnotai, L., Lin, J.L., Peng, Z.Q. and Santra, A.S., "Iterative Group Decomposition for Refining Microaggregation Solutions", ***Symmetry***, Vol. 10, No. 7, (2018), 262-274. https://doi.org/10.3390/sym10070262
18. Mortazavi, R. and Jalili, S., "Fine granular proximity breach prevention during numerical data anonymization", ***Transactions on Data Privacy***, Vol. 10, No. 2, (2017), 117-144.
19. Mortazavi, R. and Erfani, S.H., "GRAM: An efficient (k, l) graph anonymization method", ***Expert Systems with Applications***, Vol. 153, (2020), In press. https://doi.org/10.1016/j.eswa.2020.113454
20. Pelleg, D. and Moore, A.W., "X-means: Extending k-means with efficient estimation of the number of clusters", In Proceedings of the Seventeenth International Conference on Machine Learning (ICML), (2000), 727-734.
21. Ünlü, R. and Xanthopoulos, P., "Estimating the number of clusters in a dataset via consensus clustering", ***Expert Systems with Applications***, Vol. 125, (2019), 33-39. https://doi.org/10.1016/j.eswa.2019.01.074

8. APPENDIX

THE IMHM ALGORITHM

The appendix presents the pseudo-code of IMHM in brief. More details about the algorithm can be found in [9,11]. The main idea of IMHM is to calculate grouping errors incrementally to improve the time complexity of MHM [9]. The pseudo-code of the IMHM is provided in Algorithm 2. The algorithm accepts original records (as an ordered set) T, the privacy parameter k, and outputs SSE and record labels. The trivial case of $n < 2k$ is handled in Step 1 in which all records are assigned to one group. In Step 2, a directed acyclic graph $M(V, E)$ with $V = \{v_0, v_1, \dots, v_n\}$ is initialized (v_0 is a dummy node and v_i represents X_i in T for $0 < i \leq n$). In the following steps, some directed arcs are added to M. The weight of each arc equals to the SSE of grouping records between the start and end nodes of the arc. In Steps 6-7, the $Centroid$ and SSE of the first group are calculated. The results are stored in Steps 8-9 for later usage. Then, other records are added to the cluster until the size of the group reaches the limit of $2k - 1$ or no more record remains for addition. The weight of each arc is computed progressively in Steps 11-21. In Step 22, the shortest path from v_0 to v_n is saved in SP. The microaggregation error SSE and assignment A are computed in Steps 23 and 24-28, respectively. Finally, the values are returned in Step 29.

Algorithm 2. The pseudo-code of the Improved MHM (IMHM) [11]

Input: $T = (X_1, X_2, \dots, X_n)$: dataset of original records in order, k: the privacy parameter

Output: SSE: microaggregation error, A: the assignment of records to clusters with at least k members

```
1   **If** $n < 2k$, assign all records to the same cluster, calculate its $SSE$ and **Return** $SSE$ and the assignment
2   Initialize the directed acyclic graph $M(V,E), |V| = n + 1$
3   **For** $i \leftarrow 0$ **To** $n - k$
4       **For** $j \leftarrow i + k$ **To** $min\ (n,\ i + 2k - 1)$
5           **If** $i = 0$ **And** $j = i + k$
6               $CurrentCentroid \leftarrow$ MEAN ($X_1$ to $X_k$)
7               $CurrentSSE \leftarrow$ calculate $SSE$
8               $BaseCentroid \leftarrow CurrentCentroid$
9               $BaseSSE \leftarrow CurrentSSE$
10          **Elseif** $j = i + k$
11              $Delta \leftarrow X_{i+k} - X_i$
12              $CurrentCentroid \leftarrow BaseCentroid + Delta/k$
13              $CurrentSSE \leftarrow BaseSSE + \frac{\sum_{l=1}^{d} Delta[l].\left((1-k)Delta[l] + 2k(X_{i+k}[l] - CurrentCentroid[l])\right)}{k}$
14              $BaseCentroid \leftarrow CurrentCentroid$
15              $BaseSSE \leftarrow CurrentSSE$
16          **Else**
17              $s \leftarrow j - i$        // the group size
18              $OldCentroid \leftarrow CurrentCentroid$
19              $CurrentCentroid \leftarrow OldCentroid + (X_j - OldCentroid)/s$
20              $CurrentSSE \leftarrow CurrentSSE + \sum_{l=1}^{d}(X_j[l] - CurrentCentroid[l])(X_j[l] - OldCentroid[l])$
21          Draw a directed edge $e = (v_i, v_j)$ and set the weight $w(v_i, v_j) \leftarrow CurrentSSE$.
22  Compute $SP$ as the shortest path from $v_0$ to $v_n$ in $M(V,E)$
23  $SSE \leftarrow$ The length of $SP$
24  $ClusterCounter \leftarrow 1$
25  **Foreach** edge $e = (v_i, v_j) \in SP$
26      **Foreach** $v_m,\ i < m \leq j$
27          Assign $X_m$ to $G_{ClusterCounter}$       // $A[m] \leftarrow ClusterCounter$
28      $ClusterCounter \leftarrow ClusterCounter + 1$
29  **Return** $SSE$ and $A$
```

Persian Abstract

چکیده

خوشه‌بندی یکی از روش‌های اصلی در داده کاوی است که کاربردهای فراوانی در علوم کامپیوتری، زیست شناسی و علوم اجتماعی دارد. خوشه‌بندی مقید نوعی خوشه‌بندی است که در آن اطلاعات اضافی ارائه شده توسط کاربر در طی خوشه‌بندی دخالت داده می‌شود. یکی از انواع الگوریتم‌های مورد پژوهش در زمینه خوشه‌بندی مقید، الگوریتم زیرتجمیع است. در ریزتجمیع، الگوریتم خوشه بندی باید مجموعه داده را به گروه هایی با حداقل k عضوتقسیم کند که k یک پارامتر تعریف شده توسط کاربر است. کاربرد اصلی ریزتجمیع در کنترل افشای آماری است که برای انتشار داده‌ها با حفظ حریم خصوصی کاربرد دارد. کیفیت الگوریتم ریزتجمیع بر اساس مجموع مربع خطاهای داخل گروه اندازه گیری می‌شود. متاسفانه ثابت شده که ریزتجمیع بهینه در حالت کلی یک مسئله بهینه سازی NP-سخت است، اما نسخه تک بعدی آن به صورت بهینه و در زمان چند جمله‌ای قابل حل است. روش های ابتکاری زیادی براساس تبدیل به نسخه تک متغیره پیشنهاد شده است. این روشها باید داده‌ها چند بعدی را در یک دنباله مرتب کنند. این مقاله روش جدیدی برای مرتب سازی داده ها پیشنهاد می‌کند. با شروع از یک خوشه بندی اولیه، روش پیشنهادی به صورت تکراری رکوردهای چند بعدی را در یک دنباله مرتب کرده و آنها را خوشه‌بندی می‌کند. این فرایند تا زمانی که بهبودی حاصل نشود ادامه می‌یابد. مجموعه وسیعی از آزمایش‌های تجربی انجام شده در این تحقیق نشان از برتری روش پیشنهادی برای پارامترها و مجموعه داده‌های مختلف دارد.