Research on Anonymization Algorithms Based on Local Partitioning (Postprint)

Authors: Wang Fang, Yu Dunhui, Zhang Wanshan

Date: 2018-09-12T00:00:00+00:00

Abstract

To address the issue where generalization incurs substantial information loss, a limitation that becomes increasingly pronounced with growing data dimensionality, we propose an anonymization algorithm based on local partitioning. While ensuring k-anonymity and l-diversity, the data table is horizontally partitioned into multiple buckets based on sensitive attribute column value constraints and inter-record distances. Each bucket is then vertically divided into several columns according to inter-attribute associations. Finally, data within each column of the same bucket is randomly shuffled. Experimental results indicate that for high-dimensional data processing, compared to the LGAA-CP algorithm, the proposed algorithm reduces information loss by 47% to 183% and improves association preservation rate by 24% to 118%. Compared to the Slicing algorithm, the difference in information loss remains within 1.5%, while the association preservation rate improves by 8.9% to 22.8%. Analysis demonstrates that the algorithm effectively ensures both privacy protection capability and data utility for high-dimensional data.

Full Text

Anonymous Algorithm Based on Local Partitioning

Wang Fang[1], Yu Dunhui[1,2]†, Zhang Wanshan[1,2] 1. College of Computer & Information Engineering, Hubei University, Wuhan 430062, China 2. Hubei Education Information Engineering & Technology Center, Wuhan 430062, China

Abstract: Generalization causes significant data information loss, a problem that becomes more pronounced as data dimensionality increases. To address this issue, we propose an anonymous algorithm based on local partitioning. While ensuring k-anonymity and l-diversity, the algorithm first horizontally divides the

data table into several buckets based on sensitive attribute column value constraints and inter-record distances. Each bucket is then vertically partitioned into multiple columns based on relationships between attributes. Finally, data within each column of the same bucket undergo random rearrangement. Experimental results demonstrate that when processing high-dimensional data, compared with the LGAA-CP algorithm, the proposed algorithm reduces information loss by 47% to 183% and improves relationship retention rate by 24% to 118%. Compared with the Slicing algorithm, information loss differs by within 1.5%, while association relationship retention rate improves by 8.9% to 22.8%. Analysis shows that the algorithm effectively ensures both privacy protection capability and data availability for high-dimensional data.

Keywords: privacy-preserving data publishing; k-anonymity; l-diversity; sensitive attribute column value constraint

0 Introduction

With the widespread use of information technology and the growing popularity of big data technology, data openness has become an inevitable trend. Since data released by governments, departments, and enterprises contains substantial privacy information, releasing data directly in its original form would inevitably lead to privacy leakage [1]. In this context, privacy-preserving data publishing (PPDP) has been proposed. Unlike traditional privacy protection methods that use data encryption and access control to prevent unauthorized access, PPDP aims to maintain the usability of published data while preventing data users from identifying individuals corresponding to sensitive information, thereby achieving privacy protection [2-3].

As a research hotspot in this field, numerous scholars have conducted extensive research, proposing many data publishing models such as k-anonymity [4,5], l-diversity [6,7], and t-closeness [8,9]. However, one problem with these studies is that while they achieve privacy protection reasonably well, they suffer from large information loss and low data availability. To address this issue, many scholars have begun to consider data availability while achieving privacy protection. In 2012, Xu et al. proposed the WAK-anonymity algorithm based on attribute weights [10], which solved the problem of large information loss for important data but increased information loss for other data, with significant loss of relationships between data. In 2012, Li et al. proposed a clustering-based algorithm for protecting sensitive data attributes [11]. In 2016, Gong Qiyuan proposed a semi-partitioning data anonymization algorithm for high-dimensional data [12]. In 2017, Liao et al. proposed a classification anonymity algorithm based on weighted attribute entropy [13]. These methods reduce information loss to some extent, but the effects are not significant. In the same year, Jiang Huowen proposed a greedy clustering anonymization method [14], achieving balanced partitioning of equivalence classes and further improving data availability, but still could not avoid the problem of large information loss for high-dimensional data.

To reduce information loss and improve data availability, new methods have been proposed successively. In 2012, Li et al. [15] introduced the Slicing algorithm concept based on random rearrangement technology, providing a solution for reducing information loss and retaining useful relationships between attributes. In 2013, Li et al. [16] proposed the Slicing algorithm, which uses horizontal bucketing and vertical column partitioning to maintain relationships between attributes to some extent while ensuring privacy protection. In 2014, Yang et al. [17] proposed an overlapping partitioning anonymity algorithm based on Slicing, which allows an attribute to be partitioned into multiple columns. Its advantage is that relationships can be better preserved, but its disadvantage is that privacy protection capability becomes insufficient when data dimensionality is very high. In 2015, Rohilla proposed the LDS algorithm [18], which only solved the problem of continuous data discretization based on Slicing. In 2018, Wang et al. proposed the T-Closeness Slicing algorithm for transaction data [19], which makes published data satisfy the stricter T-Closeness model and improves privacy protection capability, but reduces data availability. In 2016, Wang et al. proposed a privacy data publishing method based on weighted Bayesian networks [20], which improved the accuracy of original privacy-published datasets using differential privacy technology, but still had some shortcomings when processing high-dimensional data. In summary, these methods struggle to simultaneously ensure good privacy protection capability and data availability.

Therefore, this paper proposes an ASG-LS (association set generating-local slicing) anonymous algorithm based on local partitioning. First, based on the generalization hierarchy lattice formed by generalization hierarchy tree transformation, we identify the sensitive attribute vertex set S_T and the relationship attribute vertex set R_T that does not contain sensitive attributes. Then, according to S_T, we horizontally partition the data table T into multiple buckets. Next, based on R_T, we vertically partition different attributes in the same bucket to form columns. Finally, we rearrange the data in each column of the same bucket. Thus, while achieving privacy protection, we ensure that the published data has higher availability.

1 Theoretical Foundation for Anonymous Algorithms

Assume the data table T to be published has already removed identifier attributes such as name and ID number. The data table T contains attributes $A = \{A_1, A_2, \ldots, A_n\}$, where $A_i (1 \leq i \leq n)$ are quasi-identifier attributes and S are sensitive attributes.

Definition 1 (Generalization Hierarchy Tree). If any parent-child nodes in a tree satisfy the generalization relationship, the tree is called a generalization hierarchy tree. In the generalization hierarchy tree of attribute A_i, the j-th node at the k-th level is denoted as $node_{ijk}$, with value domain $D(node_{ijk})$. The symbol "$<$" represents the generalization relationship between attribute value domains. If nodes $node_{ijk}$ and $node_{ilm}$ have a parent-child relationship, then $D(node_{ijk}) < D(node_{ilm})$ must hold. Additionally, when generally referring

to a node in the generalization hierarchy tree of attribute A_i, it is denoted as $node_i$.

Definition 2 (Generalization Lattice). If any vertices connected by edges in a graph satisfy the generalization relationship, and vertices are sets of nodes from different attribute generalization hierarchy trees, the graph is called a generalization lattice. A generalization lattice formed by n different attribute generalization hierarchy trees is denoted as $g = \{node_1, node_2, \dots, node_n\}$, where any vertex v in g is a combination of nodes from these n generalization hierarchy trees. If two vertices contain node combinations where only one pair of corresponding attribute nodes has a generalization relationship while the other nodes are identical, then a directed edge is generated between the two vertices based on the generalization relationship. Moreover, any vertex v has exactly one starting vertex v_s and one ending vertex v_e satisfying $v_s < v_e$, with all other nodes in the two vertices being identical.

Definition 3 (Sensitive Attribute Column Value Constraint). The constraint conditions followed when partitioning data table T into buckets, specifically including: (a) sensitive attribute columns should satisfy l-diversity; (b) associations may exist between attributes within a sensitive attribute column, but these attributes should not have associations with other attributes outside the column.

Definition 4 (Bucket). A horizontal partition of data table T based on sensitive attribute column value constraints and distances between records. Assuming a data table T is partitioned into n buckets $\{B_1, B_2, \dots, B_n\}$, then $T = \bigcup_{i=0}^{n-1} B_i$, and for all k, j, $B_k \cap B_j = \emptyset (0 \le k, j \le n)$.

Definition 5 (Column). A bucket B in data table T can be vertically partitioned into m columns based on relationships between attributes, denoted as $C = \{C_1, C_2, \dots, C_m\}$, where $B = \bigcup_{i=0}^{m-1} C_i$, and for all j, k, $C_j \cap C_k = \emptyset (1 \le j, k \le m)$.

Definition 6 (Vertex Constraint). If attributes in record t have corresponding attribute nodes in vertex v, then the attribute values must be within the value domain of their corresponding attribute nodes, $t \in T$.

Symbol Notation

Symbol	Meaning
T	Data table
S_T	Vertex set containing sensitive attribute associations
R_T	Vertex set not containing sensitive attribute associations
F_T	Frequent generalization hierarchy tree set
g	Generalization lattice
$node_i$	Node in generalization hierarchy tree of attribute A_i
v	Vertex in generalization lattice
B	Bucket

Symbol	Meaning
C	Column
mc	Maximum number of attribute columns in a column
nc	Minimum number of attribute columns in a column
t	Record in data table
p_i	i-th categorical attribute of record t_p
q_i	i-th numerical attribute of record t_q
$confTh$	Confidence threshold
$supTh$	Support threshold

2 ASG-LS Anonymous Algorithm Based on Local Partitioning

2.1 Algorithm Overview

To ensure privacy information is not leaked while improving the retention rate of relationships between published data and thereby enhancing data availability, we propose the ASG-LS anonymous algorithm based on local partitioning for scenarios where relationships between attributes vary with value domains. The algorithm sequentially calls the ASG algorithm and LS algorithm. The ASG algorithm identifies multi-dimensional and multi-level relationships between attribute value domains and saves vertices containing these relationships in descending order of importance, storing vertices containing sensitive attributes in the sensitive attribute association set S_T and other vertices in the ordinary association set R_T. The LS algorithm implements partitioning and anonymization of the data table. Its basic idea is to horizontally divide the data table into several buckets based on sensitive attribute value constraints and distances between records, then vertically partition each bucket into multiple columns based on relationships between attributes, and finally randomly rearrange data in each column of the same bucket. This ensures that the published data maintains higher utility value while achieving privacy protection.

2.2 Attribute Association Relationship Set Generation Algorithm ASG

The ASG algorithm completes the preprocessing work for the LS algorithm, implemented based on Apriori and hierarchical generalization trees. Its purpose is to find multi-dimensional and multi-level relationships between attribute value domains, save vertices containing these relationships in descending order of importance, store vertices containing sensitive attributes in the sensitive attribute association set S_T, and other vertices in the ordinary association set R_T. The ASG algorithm is shown in Algorithm 1.

The algorithm' s main implementation process consists of four steps:

a) **Generate Frequent Generalization Hierarchy Trees.** Starting from

leaf nodes and based on breadth-first search, traverse the generalization hierarchy tree from left to right and bottom to top. Calculate each node' s support based on the probability of its attribute value domain appearing in the data table. If support $\geq supTh$, retain the node and mark directly connected nodes; otherwise, delete the node and continue traversing connected nodes. During traversal, if a node has already been marked, retain it directly. Ultimately, all nodes in the generalization hierarchy tree have support $\geq supTh$, forming frequent generalization hierarchy trees.

b) **Convert Frequent Generalization Hierarchy Trees to Generalization Lattices.** Connect frequent generalization hierarchy trees to generate generalization lattices: select two frequent generalization hierarchy trees of different attributes, connect them pairwise to generate generalization lattice g, whose vertices are combinations of nodes from the two trees. If two vertices contain node combinations where only one pair of corresponding attribute nodes has a generalization relationship while the other nodes are identical, generate a directed edge based on the generalization relationship. Connect frequent generalization hierarchy trees with frequent generalization lattices to produce generalization lattice g_i: if frequent generalization lattice g does not contain this frequent generalization hierarchy tree, connect them and generate generalization lattice g_i using the above method; otherwise, do not connect.

c) **Generate Sensitive and Non-sensitive Attribute Association Sets.** Starting from vertices with no incoming edges, based on breadth-first search, traverse and calculate the support and confidence of each vertex in generalization lattice g from bottom to top by level. If both are greater than or equal to the set thresholds, save the vertex and, based on whether it contains sensitive attributes, save it to sensitive attribute association set S_T or non-sensitive attribute association set R_T, then mark directly connected vertices; otherwise, delete the vertex. During traversal, if a vertex has already been marked, retain it directly. The final association sets save vertices in the generalization lattice that have relationships, relatively maximum vertex levels, and non-intersecting value domains for corresponding nodes. All vertices in the generalization lattice become frequent generalization lattices.

d) **Implement Vertex Importance Sorting.** To retain more useful relationships, vertices in the subsequent LS algorithm are prioritized for partitioning based on those containing important relationships. The importance $value(v)$ of vertex v is an indicator evaluating whether the relationships between attributes contained in a vertex are important. It is determined by the levels of all nodes contained in the vertex and the vertex' s confidence. First, use formula (1) to calculate the normalized level value of node $inode$. Where $lev(inode)$ is the level of node $inode$ in the generalization hierarchy tree, and $maxlev$ is the maximum level of the generalization hierarchy tree.

$$lev(node) = \frac{lev(inode) - lev(root)}{lev(max) - lev(root)} \tag{1}$$

Then, use formula (2) to calculate the importance $value(v)$ of vertex v.

$$value(v) = \sum_{i=1}^{n} w_i \cdot lev(node_i) + conf(v) \tag{2}$$

Where w_i represents the weight of attribute A_i in data table T, n is the number of nodes contained in vertex v, and $conf(v)$ is the confidence of vertex v.

Algorithm 1 ASG Algorithm

```
Input: Data table T, attribute generalization hierarchy tree Set, confidence threshold conf
       support threshold supTh, maximum attribute columns in a column mc, attribute weights
Output: Sensitive attribute association vertex set S_T, non-sensitive attribute association

1. Generate frequent generalization hierarchy tree set FT_Set from Set based on supTh
2. for each pair of frequent generalization hierarchy trees in FT_Set
3.     Connect to generate generalization lattice g and add to generalization lattice set
4. for each generalization lattice g_i in the generalization lattice set
5.     while there are vertices with no incoming edges
6.         Enqueue vertices by level
7.     while queue is not empty
8.         Dequeue vertex v
9.         if v is not marked
10.            Calculate support and confidence of vertex v in generalization lattice g_i
11.            if support   supTh and confidence   confTh
12.                Save vertex v to S_T or R_T based on whether it contains sensitive attrib
13.                Mark vertices directly connected to v
14.            else
15.                Delete vertex v, enqueue vertices directly connected to v
16.            end if
17.        else
18.            Add generalization lattice g_i to frequent generalization lattice set
19.        end if
20.    end while
21.    Connect each generalization lattice g_i in the set with each frequent generalization
       add to new generalization lattice set
22. end for
23. Iterate step 3 until vertices in generalization lattice contain mc nodes
24. Calculate importance of each vertex based on attribute weights w
25. Sort vertices in S_T and R_T in descending order of importance
26. return S_T, R_T
```

2.3 LS Algorithm Based on Local Partitioning

The LS algorithm implements data table anonymization based on Slicing concepts and association relationship sets. Its basic idea is to first horizontally partition the data table into multiple buckets, then vertically partition each bucket into multiple columns. The result is that the same attribute may not be in the same column across different buckets. This ensures both reduced information loss and reduced loss of relationships between data.

The algorithm' s main implementation process consists of four steps:

a) **Horizontal Partitioning into Buckets.** To reduce data information loss, the data table is partitioned into buckets. Each time, the vertex v with the maximum importance value is taken from S_T, then a record t satisfying the vertex v value constraint is randomly selected from data table T and added to bucket B. Subsequently, $k-1$ records satisfying sensitive attribute column value constraints and nearest distance constraints are selected from data table T into bucket B.

To select records satisfying the nearest distance constraint, the distance between any record in the data table and record t must be calculated, measured by the distance between attributes of the two records. Attribute distances are divided into numerical attribute distances and categorical attribute distances. The distance between the i-th numerical attribute of record t_p and the i-th numerical attribute of record t_q is $dist_{nn}(t_p.A_i, t_q.A_i)$. The distance between the i-th categorical attribute of record t_p and the i-th categorical attribute of record t_q is $dist_{cc}(t_p.A_i, t_q.A_i)$.

The distance between numerical quasi-identifier attributes is calculated as follows, where $D(t_p.A_i)$ represents the value of the i-th numerical quasi-identifier of tuple t_p, $|D(t_p.A_i)-D(t_q.A_i)|$ is the absolute difference between two attribute values, $max_i(T.A)$ represents the maximum value of the i-th numerical identifier in data table T, and $min_i(T.A)$ represents the minimum value.

$$dist_{nn}(t_p.A_i, t_q.A_i) = \frac{|D(t_p.A_i) - D(t_q.A_i)|}{max_i(T.A) - min_i(T.A)} \tag{3}$$

The distance calculation between categorical quasi-identifier attributes requires introducing the generalization hierarchy tree as a measurement basis. First, calculate the distance from the i-th categorical attribute c_{p_i} of tuple t_p and the i-th categorical attribute c_{q_i} of tuple t_q to their common minimum ancestor node, denoted as $l(c_{p_i}, c_{q_i})$. Then calculate the distances from c_{p_i} and c_{q_i} to the root node $l(c_{p_i}, root)$ and $l(c_{q_i}, root)$. Finally, calculate the distance between categorical quasi-identifier attributes using the following formula:

$$dist_{cc}(t_p.A_i, t_q.A_i) = \frac{l(c_{p_i}, c_{q_i})}{l(c_{p_i}, root) + l(c_{q_i}, root)} \tag{4}$$

Assuming a record has d_1 numerical quasi-identifiers and d_2 categorical quasi-identifiers, the distance between records is:

$$dist(t_p, t_q) = \sum_{i=1}^{d_1} dist_{nn}(t_p.A_i, t_q.A_i) + \sum_{i=1}^{d_2} dist_{cc}(t_p.A_i, t_q.A_i) \tag{5}$$

b) **Column Partitioning.** To reduce loss of relationships between data, each bucket is vertically partitioned into multiple columns. The idea is that for each bucket, attributes corresponding to the node set contained in vertex v with sensitive attributes are first partitioned into one column. Then, each vertex v in the non-sensitive attribute association set R_T is traversed to calculate the number of records in the bucket satisfying vertex v constraints. After sorting vertices in R_T in descending order of record count, vertices are taken sequentially from front to back. For each vertex v, identify the attributes in the bucket corresponding to each node contained in v, and remove attributes that have already been assigned. If the number of remaining attributes is greater than or equal to the set minimum number of attributes in a column, partition them into one column; otherwise, abandon the partitioning. Finally, if the number of remaining unpartitioned attribute columns in the bucket is greater than the maximum number of attributes in a column, uniformly partition the attributes into columns; otherwise, directly use the remaining attributes as one column.

c) **Data Rearrangement.** Assuming attributes in a bucket are partitioned into n columns, randomly rearrange any $n-1$ columns in the bucket. After rearrangement, if any record remains in its original position, randomly swap it with other records in the bucket. Thus, the relationships between highly associated attributes in the same column remain unchanged, while relationships between different columns are disrupted, thereby preserving relationships between highly associated attributes.

d) **Column Generalization.** To make data table T satisfy k-anonymity, check the probability of an attribute value appearing in data table T within a bucket. If below threshold p_t, generalize the column containing that attribute. After column generalization, calculate the number of columns where the number of distinct records equals k. If the number of such columns is less than 2, generalize the entire bucket.

Algorithm 2 LS Algorithm

```
Input: Data table T, sensitive attribute association vertex set S_T, non-sensitive attribute
       association vertex set R_T, k-anonymity parameter k, l-diversity parameter l,
       maximum attributes in a column mc, minimum attributes in a column nc
Output: Anonymized data table T*

1. while T is not empty and there are unprocessed records in T
```

```
    Take vertex v from S_T in order from front to back
    while number of records satisfying vertex v constraints > 0
        Randomly take a record t from T satisfying vertex v constraints, add to bucket B
        while number of records in bucket B < k
            Take the record closest to t and satisfying sensitive attribute column value
             constraints, add to bucket B
        end while
        Partition attributes corresponding to the node set contained in vertex v into one
        for each vertex v_i in R_T
            Calculate number of records n_i in bucket B satisfying vertex v_i constraints
        end for
        Sort vertices in R_T in descending order of record count n_i
        for each vertex v_i in R_T in order from front to back
            Find attributes in bucket B corresponding to each node in v_i,
             remove attributes already assigned to other columns
            if remaining attribute columns > nc
                Partition them into one column
            else
                Abandon this partitioning
            end if
        end for
        if remaining unpartitioned attribute columns in bucket B > mc
            Uniformly partition attributes into columns
        else
            Use remaining attributes directly as one column
        end if
        if bucket B is partitioned into n columns total
            Randomly rearrange n-1 columns
        end if
        After rearrangement, if any record remains in original position,
         randomly swap it with other records in bucket B
        Check probability of attribute values in bucket B appearing in data table T
        if probability below threshold p_t
            Generalize the column containing that attribute
        end if
        After column generalization, calculate number of columns where distinct records =
        if number of columns < 2
            Generalize entire bucket
        end if
    end while
end while
return T*
```

2.4 Time Complexity Analysis

The ASG-LS anonymous algorithm consists of two sub-algorithms: ASG and LS. The ASG algorithm generates frequent generalization hierarchy trees and converts them to generalization lattices, both with time complexity $O(n^2)$. The other two steps also have time complexity $O(n^2)$. Therefore, the time complexity of sub-algorithm ASG is $O(n^2)$. The time complexity of the LS algorithm only relates to the number of vertices in S_T, the number of vertices in R_T, the k-anonymity parameter k, and the number of records n in table T. The number of vertices and k value are much smaller than n, so the time complexity is $O(n^2)$. Thus, the total time complexity of the ASG-LS algorithm is $O(n^2)$.

3 Experimental Analysis

3.1 Algorithm Comparison

We compare the proposed ASG-LS algorithm with two types of methods: generalization and Slicing. Based on original patient diagnosis records from a hospital, as shown in , we compare the three algorithms.

After processing the data in with generalization, the result is shown in . The result satisfies 4-anonymity and 3-diversity, providing a certain level of privacy protection for the published data table. However, excessive generalization leads to excessive information loss and low data availability.

After processing with the Slicing algorithm, the result is shown in . The algorithm partitions sensitive attributes and related attributes into one column and other attributes into another column, achieving privacy protection through column data permutation. The result satisfies 3-diversity and 4-anonymity, achieving a certain level of privacy protection. Compared with generalization, Slicing performs both horizontal and vertical partitioning, thereby improving data availability. However, its disadvantage is that relationships between attributes vary with value domains, losing some associations between attributes.

After processing with our ASG-LS algorithm, the result is shown in . The entire table is divided into two buckets, each partitioned into two columns. The results clearly satisfy 4-anonymity and 3-diversity, meeting privacy protection requirements well. Unlike Slicing where the same attribute must be in the same column across different buckets, ASG-LS uses local partitioning to assign the same attribute to different columns in different buckets, preserving more relationships between attributes and thus significantly improving data availability.

3.2 Experimental Analysis

We evaluate the performance of the ASG-LS algorithm from two aspects: information loss [13] and data availability. GAA-CP [14] is a generalization algorithm with small information loss, representing excellent generalization algorithms. However, GAA-CP only satisfies k-anonymity; for better comparison,

we modify it to satisfy l-diversity, denoted as LGAA-CP. The k value in the Slicing [16] algorithm represents the minimum number of records in a bucket.

The experimental dataset comes from the Adult dataset, widely used in privacy protection research. After removing records with missing attribute values, the experiments were completed based on a dataset containing 30,162 tuples. The experimental environment: Intel(R) CoreTM i5-2450M CPU @2.50GHz; 4.00 GB memory; LITEON T9 (256 GB) primary hard drive; Windows 10 Professional 64-bit operating system; MySQL database system; IntelliJ IDEA 2017.2.5 development environment; jdk8 runtime environment. The algorithm was implemented in Java.

Information Loss Analysis To analyze how data information loss changes with dataset dimensionality $|T_m|$, anonymity parameter k value, we conducted two groups of experiments using ASG-LS, LGAA-CP, and Slicing algorithms.

Experiment 1: When data dimension $|T_m|$ is 5, 10, and 14 respectively, verify the impact of anonymity parameter k value changes on information loss. Results are shown in Figure 1: see original paper(b)(c). Where maximum attributes in a column $mc = 4$, minimum attributes in a column $nc = 2$, confidence threshold $confTh = 0.8$, support threshold $supTh = 0.2$, and l-diversity parameter $L = 3$.

The experimental results show that when data dimension is the same, as k value increases, information loss increases for all three algorithms, but LGAA-CP' s information loss increases significantly. Slicing and ASG-LS show minimal information loss increase, and the two are basically consistent because both perform local generalization.

Experiment 2: When anonymity parameter k is 10, 15, and 20 respectively, verify the impact of data dimension $|T_m|$ changes on information loss. Results are shown in Figure 2: see original paper(b)(c). Where maximum attributes in a column $mc = 3$, minimum attributes in a column $nc = 2$, confidence threshold $confTh = 0.8$, support threshold $supTh = 0.2$, and l-diversity parameter $L = 3$.

The experimental results show that when anonymity parameter k is the same, as data dimension $|T_m|$ increases, information loss increases for all three algorithms. However, LGAA-CP' s information loss increases significantly, with the fastest growth when $|T_m|$ is between 5 and 8. Since both Slicing and ASG-LS perform local generalization, their information loss is smaller and increases more slowly.

Association Relationship Retention Analysis Association relationship retention rate AR is the ratio of association rules retained after data anonymization to the number of association rules in original data, measuring the effectiveness of anonymization methods in preserving useful relationships between attributes. This paper uses Weka Apriori to mine association rules.

To analyze how association relationship retention rate changes with dataset dimensionality $|T_m|$ and k-anonymity parameter k, we conducted two groups of

experiments using ASG-LS, LGAA-CP, and Slicing algorithms.

Experiment 3: When data dimension $|T_m|$ is 5, 10, and 14 respectively, verify the impact of anonymity parameter k value changes on association relationship retention rate. Results are shown in Figure 3: see original paper~(c). Where maximum attributes in a column $mc = 4$, minimum attributes in a column $nc = 2$, confidence threshold $confTh = 0.8$, support threshold $supTh = 0.2$, and l-diversity parameter $L = 3$.

The experimental results show that when data dimension $|T_m| = 5$, the three algorithms have the same association relationship retention rate because when dimension is small, original data has fewer association rules, and much data already satisfies k-anonymity by its own characteristics, so algorithm dependency is not strong. When data dimensions are 10 and 14, as k value increases, LGAA-CP' s association relationship retention rate decreases significantly, while ASG-LS and Slicing show slight decreases, but ASG-LS always remains above Slicing because both perform local generalization, and ASG-LS performs local partitioning.

Experiment 4: When anonymity parameter k is 10, 15, and 20 respectively, verify the impact of data dimension $|T_m|$ changes on association relationship retention rate. Results are shown in Figure 4: see original paper(b)(c). Where maximum attributes in a column $mc = 3$, minimum attributes in a column $nc = 2$, confidence threshold $confTh = 0.8$, support threshold $supTh = 0.2$, and l-diversity parameter $L = 3$.

The experimental results show that all three algorithms experience rapid decreases in association relationship retention rate when data dimension $|T_m|$ is between 5 and 8 because the number of association relationships in original data increases significantly when dimension increases from 5 to 8. When data dimension increases from 8 to 14, all three algorithms show gradual decreases in association relationship retention rate, but ASG-LS' s retention rate is greater than Slicing and LGAA-CP due to its small local generalization and local partitioning.

4 Conclusion

To ensure privacy information is not leaked while improving data availability after anonymization, this paper proposes an anonymous algorithm based on local partitioning. Building on Slicing concepts and ensuring k-anonymity and l-diversity, the algorithm horizontally divides the data table into several buckets based on sensitive attribute column value constraints and inter-record distances, then vertically partitions each bucket into multiple columns based on relationships between attributes, and finally randomly rearranges data in each column of the same bucket. Experimental results demonstrate that the proposed method has high rationality and effectiveness in reducing information loss and improving association relationship retention rate.

References

[1] Wu Yi, Wang Dan, Jiang Zongli. Privacy preserving in re-publication of dynamic set-valued data based on transactional k-anonymity [J]. Journal of Computer Research and Development, 2013, 50(s1): 248-256.

[2] Feng Dengguo, Zhang Ming, Li Hao. Big data security and privacy protection [J]. Chinese Journal of Computers, 2014, 37(1): 246-258.

[3] Liu Yahui, Zhnag Tieying, Jin Xiaolong. Personal privacy protection in the era of big data [J]. Journal of Computer Research and Development, 2015, 52(1): 229-247.

[4] Latanya S. k-anonymity: a model for protecting privacy [J]. International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, 10(5): 557–570.

[5] Dong Fangfei. Research on privacy protection algorithm based on k-anonymity [D]. Lanzhou: Northwest Normal University, 2015.

[6] Machanavajjhala A, Kifer D, Gehrke J. l-diversity: privacy beyond k-anonymity [C]// Proc of International Conference on Data Engineering. 2006: 24.

[7] Yang Gaoming, Li Jingzhao, Zhang Shunxiang, et al. An enhanced l-diversity privacy preservation [C]// Proc of International Conference on Fuzzy Systems and Knowledge Discovery. 2014: 1115-1120.

[8] Li Ninghui, Li Tiancheng, Venkatasubramanian S. t-closeness: privacy beyond k-anonymity and l-diversity [C]// Proc of International Conference on Data Engineering. IEEE, 2007: 106-115.

[9] Zhang Jianpei, Xie Jing, Yang Jing, et al. A T-Closeness privacy model based on sensitive attribute values semantics bucketization [J]. Journal of Computer Research and Development, 2014, 51(1): 126-137.

[10] Xu Yong, Qin Xiaolin, Yang Yitao, et al. A weight-aware approach to privacy preserving publishing data set [J]. Journal of Computer Research and Development, 2012, 49(5): 913-924.

[11] Li Shanshan, Zhu Yuquan, Chen Geng. Clustering-based algorithm for data sensitive attributes anonymous protection [J]. Application Research of Computers, 2012, 29(2): 469-471.

[12] Gong Qiyuan. Research on data anonymous technology for data publishing [D]. Nanjing: Southeast University, 2016.

[13] Liao Jun, Jian Chaohui, Guo Chun, et al. Classification anonymity algorithm based on weight attributes entropy [J]. Computer Science, 2017, 44(7): 42-46.

[14] Jiang Huowen, Zeng Guosun, Ma Haiying. Greedy Clustering-Anonymity Method for Privacy Preservation of Table Data-publing [J]. Journal of Software, 2017, 28(2): 341-351.

[15] Li Tiangcheng, Li Ninghui, Zhang Jian, et al. Slicing: A New Approach for Privacy Preserving Data Publishing [J]. IEEE Trans on Knowledge & Data Engineering, 2012, 24(3): 561-574.

[16] Li Tiangcheng, Li Ninghui, Zhang Jian, et al. Slicing: a new approach to privacy preserving data publishing [J]. International Journal of Computer Trends & Technology, 2013, 4(8): 64-78.

[17] Yang Jing, Liu Ziyun, Yang Yue, et al. A data anonymous method based on overlapping slicing [C]// Proc of International Conference on Computer Supported Cooperative Work in Design. 2014: 124-128.

[18] Bhardwaj M, Rohilla S. Privacy preserving data publishing through slicing [J]. American Journal of Networks and Communications, 2015, 4(3-1): 45-48.

[19] Wang Mingzheng, Jiang Zhengrui, Zhang Yu, et al. t-closeness slicing: a new privacy preserving approach for transactional data publishing [J]. Social Science Electronic Publishing, 2018, 29(7): 50-63.

[20] Wang Liang, Wang Weiping, Meng Dan. Privacy preserving data publishing via weighted bayessian networks [J]. Journal of Computer Research and Development, 2016, 53(10): 2343-2353.

Note: Figure translations are in progress. See original paper for figures.