k^m-Anonymity for Continuous Data Using Dynamic Hierarchies

Olga Gkountouna[1], Sotiris Angeli[1], Athanasios Zigomitros[3,2], Manolis Terrovitis[2], and Yannis Vassiliou[1]

[1] National Technical University of Athens, Greece
olga@dblab.ece.ntua.gr, el08626@mail.ntua.gr, yv@ece.ntua.gr
[2] Institute for the Management of Information Systems (IMIS), Athens, Greece
mterrovitis@imis.athena-innovation.gr
[3] University of Piraeus, Greece
azigomit@imis.athena-innovation.gr

Abstract. Many organizations, enterprises or public services collect and manage personal data of individuals. These data contain knowledge that is of substantial value for scientists and market experts, but carelessly disseminating them can lead to significant privacy breaches, as they might reveal financial, medical or other personal information. Several anonymization methods have been proposed to allow the privacy preserving sharing of datasets with personal information. Anonymization techniques provide a trade-off between the strength of the privacy guarantee and the quality of the anonymized dataset. In this work we focus on the anonymization of sets of values from continuous domains, e.g., numerical data, and we provide a method for protecting the anonymized data from attacks against identity disclosure. The main novelty of our approach is that instead of using a fixed, given generalization hierarchy, we let the anonymization algorithm decide how different values will be generalized. The benefit of our approach is twofold: a) we are able to generalize datasets without requiring an expert to define the hierarchy and b) we limit the information loss, since the proposed algorithm is able to limit the scope of the generalization. We provide a series of experiments that demonstrate the gains in terms of information quality of our algorithm compared to the state-of-the-art.

Keywords: Privacy-Preserving Data Publishing, Privacy, k^m-anonymity, Continuous data

1 Introduction

Datasets that contain sets of numerical data are frequent in various domains. They might describe readings from sensors or from human observation, they might represent health indicators, e.g., measurements of blood pressure, or financial data, e.g., payments.

Consider the example of Table 1, which depicts payments performed by different users for a service, e.g., recharges of a transport card. If this dataset is published, then an attacker who has partial knowledge of a record might be able to identify the record

Name	*Payments*
John	{11000, 11000, 20000, 40000, 40000}
Mary	{11000, 30500, 40000}
Nick	{11000, 11000, 40000, 40000}
Sandy	{11000}
Mark	{20000}

Table 1. Original Payment data

in the published dataset. For example, Alice may know that John has made a payment of 11,000 and another one between 18,000 and 22,000. Even if names and unique identifiers are removed from the published Table 1, Alice will be be able to identify John's record in the dataset.

In this work we aim at providing protection against identity disclosure, i.e., to prevent attackers from associating a record in the published dataset with a real person. We ensure the preservation of user privacy in the published data, by guaranteeing k^m-anonymity [1]. k^m-anonymity ensures that any attacker who knows up to m items of a target record cannot use that knowledge to identify more than k individuals in the dataset. This guarantee is a relaxation of the classic k-anonymity [2]. Consider the 2^2-anonymous Table 2 which is an anonymization of Table 1. Any attacker with partial knowledge of up to 2 values of a target, will not be able to identify less than 2 records. To achieve this level of privacy in our dataset, using the data hierarchy of Figure 1, all values had to be generalized because values {20,000} and {30,500} were rare. However, the same privacy can be ensured in Table 3 where values {20,000} and {30,500} are generalized to the range [20,000-30,500]. As we can observe, less values are generalized and a smaller information loss is achieved.

The basic novelty of our method is that we do not assume a fixed generalization hierarchy, i.e., an a priori defined hierarchical mapping of the initial domain values to generalized values, but the anonymization algorithm dynamically explores different possible ways to anonymized the original domain. It relies on clustering values that lie closely together and replacing them by the smallest possible range. The benefits of our approach are twofold: a) the anonymization process does not need a clearly defined hierarchy, which can be a burden for the data publisher and b) by exploring a greater solution space, e.g., many different generalization hierarchies, it manages to significantly limit the information loss due to the anonymization.

Id	*Payments*
1	(10000-20000], (10000-20000], (30000-40000], (30000-40000], (30000-40000]
2	(10000-20000], (30000-40000], (30000-40000]
3	(10000-20000], (10000-20000], (30000-40000], (30000-40000]
4	(10000-20000]
5	(10000-20000]

Table 2. 2^2-anonymous table using a data generalization hierarchy

Id	*Payments*
1	11000, 11000, [20000-30500], 40000, 40000
2	11000, [20000-30500], 40000
3	11000, 11000, 40000, 40000
4	11000
5	[20000-30500]

Table 3. 2^2-anonymous table using a dynamic hierarchy

Our work differs from existing algorithms for k^m-anonymity [1, 3, 4] because a) it focuses on continuous values, and not categorical ones as previous approaches, b) it allows for duplicates in records, i.e., records have bag instead of set semantics and c) it does not consider a given hierarchy.

Our main contributions include the following:

- We extend the problem of anonymizing set-valued data [1] to collections of itemsets with continuous values;
- We present the main differences and challenges of applying k^m-anonymity guarantee to our data scenario;
- We propose a utility-preserving k^m-anonymization algorithm for continuous data that does not use a fixed generalization hierarchy;
- We evaluate our methods with real-world data and compare our results to the apriori algorithm of [1], a k^m-anonymity algorithm using pre-defined data generalization hierarchies for set-valued data.

The rest of the paper is organized as follows: Section 2 describes the problem and presents the attack models. In Section 3 we describe our algorithm and the data structures we use. Section 4 presents the experimental evaluation. Section 5 describes related work and in Section 6 we express our conclusions and possible future directions of this work.

2 Problem Definition

Let dataset D be a collection of records t, where each record is a collection of values v from a continuous domain $\mathscr{I}$. We assume that each record describes a different real world entity (person).

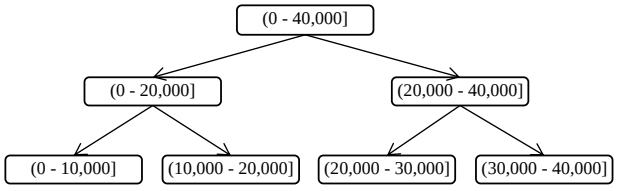

Fig. 1. Data Generalization Hierarchy of the data in Table 1.

We assume attackers that only have partial knowledge of a record, i.e., m values that are associated with a real person, and want to identify the whole record in the published data. We do not make the distinction between sensitive attributes and quasi-identifiers. Every value is a potential quasi-identifier, and all values are equally sensitive as well. The k^m-anonymity [1] guarantee is defined as follows:

Definition 1. *(k^m-anonymity guarantee [1]) A dataset D is considered k^m-anonymous if any attacker knowing up to m values of a record $t \in D$, is not able to use this knowledge to identify less than k records in D.*

k^m-anonymity requires that each record in the dataset is indistinguishable from at least $k-1$ others with reference to every possible m-sized combination of its values. In other words, any attacker who knows of m values that are associated with a person will always find k records in the published dataset that match her background knowledge. Unlike traditional k-anonymity, we do not require that records are identical. In the context of sparse multidimensional data, this would introduce great information loss, but it would also be less significant; it would protect from attackers who know a complete or almost complete record, which is unnecessary, and it would also protect against attackers who have negative knowledge, i.e., those who know that a value does not appear in a record. Negative knowledge is a weak quasi identifier in the case of sparse data and it is not covered by k^m-anonymity to increase the quality of the anonymized dataset.

A dataset D which is not k^m-anonymous, can be transformed to k^m-anonymous dataset $D^\star$, by recoding the values so that $D^\star$ satisfies the *k^m-anonymity* guarantee. To achieve this, we generalize only those values that are necessary to make every m-sized combination appear in at least k records, as in Table 3. A generalization is a set of rules in the form $v \rightarrow [a, b]$, which map a value v of the original data to a range that includes it. In this work we use global recoding, i.e., when a value a is generalized to a value A, then *all* appearances of value a in the dataset are replaced by A.

There may be many possible anonymizations of a dataset that satisfy k^m-anonymity for a given attacker's knowledge limit m, as shown in Tables 3 and 2. The worst-case scenario would be to anonymize all values to the maximum domain range $\mathscr{I}$. Such a solution is possible, but it would introduce the highest information loss and the released data would practically have no utility.

The problem of finding the optimal k^m-anonymization is to find the set of generalizations that satisfy k^m-anonymity and produce the least information loss.

3 Anonymization algorithm

3.1 Solution Space

The solution space is the set of all possible generalizations. These are all the possible substitutions of any data value v with a range that contains it. The range can be any subrange of the domain $\mathscr{I}$. The accepted solutions are those who do not violate k^m-anonymity. The problem of optimal multidimensional k-anonymity was proven to be NP-hard [5]. As we mentioned earlier, our dataset can be represented as a sparse multidimensional table, while the solution space is much larger than that of k-anonymity.

There are two reasons for this; (i) k^m-anonymity does not need to form equivalence classes where all records have identical attribute values and (ii) we do not use a generalization hierarchy, therefore the set of possible generalizations is significantly larger. To deal with the complexity of the optimal anonymization problem we have opted for a heuristic solution. We take advantage of the *apriori* principle, and perform global-recoding generalization on the infrequent values at each step of our algorithm, as we explain below.

3.2 Dynamic Count Tree

According to the a priori principle, given a frequency threshold k, any itemset of size n cannot have frequency higher than k if any of its subsets is infrequent. Equivalently, if an itemset of size n has frequency lower than k, then all its supersets of sizes $n+1, n+2$, etc. are also infrequent.

To exploit this property, our algorithm uses a tree structure similar to the FP-tree of [6]. Every node corresponds to a data value; either original or a generalized range of values. Nodes at the first level of the tree trace the *support* of the values, i.e. the number of tuples that contain the value. A path from the root to a node with depth i, corresponds to an itemset combination of values of size i. Every node n_i in an intermediate level i holds the *support* of the combination of values that appear in the path from the root to n_i.

Definition 2. (support) *The support of a combination of values in a dataset is the number of records that contain this combination.*

Sibling nodes are sorted by their support in descending order, i.e. more frequent nodes appear first. At the first step, a node for every value that appears in the dataset is added to the first level of the tree, as shown in Figure 2(a). At the next step, a new level of nodes is introduced to the count tree. These are the itemsets of size 2. Itemsets are also sorted by their support. Therefore, if the value v_1 of node n_1 is more frequent than v_2 of n_2, we expect to find the 2-itemset $\{v_1, v_2\}$ in the path $n_1 \rightarrow n_2$. At each step i of our algorithm, a new level of nodes is introduced to the count tree. Combinations with common prefixes share a common sub-path in the tree. For instance, itemsets {5, 10, 2} and {5, 10, 1} will share the path 5→10 in the tree. Note that since we allow for duplicate values in a record, nodes with the same value can appear in the same path.

The goal is for every m-sized combination of values to have support at least k. To achieve this, following the *apriori* principle, we progressively examine itemsets of sizes i=1, 2, ..., m. At each step i, we ensure that the supports of every i-itemset is at least k, before we proceed to step i+1.

3.3 Information Loss

To estimate the loss of utility introduced by the value generalizations we use the Normalized Certainty Penalty (NCP) metric [7]. Let v be a value in original domain $\mathscr{I}$. Then:

$$NCP(v) = \begin{cases} 0, & v \; is \; not \; generalized \\ |g_{max} - g_{min}|/|\mathscr{I}|, & otherwise \end{cases}$$

Algorithm 1 Incremental Creation of the Dynamic Count Tree **UpdateDCTree**

Require: D {Original Dataset}, T_{i-1} {tree of size $i-1$}, G {current generalizations}
Ensure: T_i is the count tree of height i.

```
1: for every record t ∈ D do
2:    for every value v ∈ t do
3:       if ∃ generalization range g ∈ G, such that v ∈ g then
4:          replace v with g.
5:    for every combination cmb_i of i values in t do
6:       find path p_{i-1} that contains (i-1)-subset of cmb_i (prefix)
7:       if the i^th value exists as a leaf then
8:          increase its support by 1.
9:       else
10:         add the remaining i^th value as a leaf under p_{i-1}
11: return D*
```

where $[g_{min}, g_{max}]$ is the range to which v is generalized.

The total information loss of an anonymous dataset D^* with $|D^*|$ records, is the average NCP of all its values:

$$NCP(D^*) = \frac{\sum_{t_i \in D^*} \{\sum_{v_{i,j} \in t_i} NCP(v_{i,j})\}}{\sum_{t_i \in D^*} |t_i|}$$

where $v_{i,j}$ is the j^{th} value in the i^{th} record and $|t_i|$ is the size of record the i^{th} record.

3.4 Algorithm

We propose a heuristic global-recoding generalization algorithm. As shown in the pseudo-code of Algorithm 2, our method has m basic steps. At each step $i = 1, ..., m$, our algorithm, ACD, checks for privacy violations of itemsets of size i. To check every possible i-sized combination of values, we use the count tree created by Algorithm 1.

Each path from the root to a leaf corresponds to an itemset whose support is equal to the support of that leaf. If a leaf has support less than k, then this value combination is rare and is considered vulnerable. To protect individuals whose records contain this itemset, one or more values need to be generalized. The goal is to increase the paths' support. The only way to achieve this is by generalizing a value enough, so that its generalization range will include other values belonging to sibling nodes, thus merging

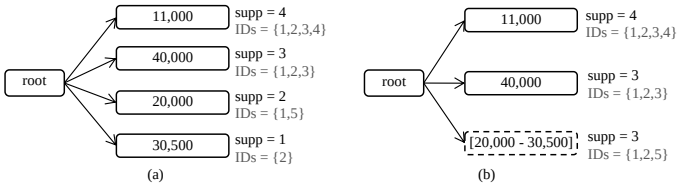

Fig. 2. (a) Count tree T_1 for the dataset of Table1. (b) T_1 after the necessary generalization 30,500→[20,000-30,500].

Algorithm 2 k^m-Anonymization of Continuous Data algorithm **ACD**

Require: D {Original Dataset}, m {maximum size of attacker's knowledge}, k {privacy parameter}, d {NCP threshold}
Ensure: $D^\star$ is a k^m-anonymous Dataset.
1: sort tuples' values with reference to their support.
2: $G = \emptyset$
3: $T_0 = null$
4: **for** i = 1, 2, ..., m **do**
5: $T_i = UpdateDCTree(D, T_{i-1}, G)$
6: **for** every leaf node f in T_i **do**
7: **if** $support(f) < k$ **then**
8: $G_f = findGeneralizations(T_i, f, k, d)$
9: add generalization rules: $G = G \cup G_f$.
10: parse T_i in a breadth-first traversal
11: **if** there exist sibling nodes with values $v_1, ..., v_n \in g$, where $g \in G_f$ **then**
12: replace values $v_1, ..., v_n$ with g
13: merge them into a single node n
14: update n's support
15: **return** $D^\star$

the node with one of its siblings and combining their supports. If the siblings' values appear in different records, then the support of the merged node will be higher than the supports of the initial nodes. The merged node's value will be the minimum range that includes the initial values.

Since we opt for global-recoding, once a generalization rule $v \rightarrow [v_{min}, v_{max}]$ is decided by the algorithm for a value v, then every other value v', such that $v' \in [v_{min}, v_{max}]$, will also be generalized to the same range in the dynamic count tree, as shown in lines 10-14 of Algorithm 2. This causes siblings whose values fall in the same generalization range to be merged together. This happens for nodes in all levels of the tree as well, thus reducing the tree's size. Every generalization that has been decided in the previous steps 1,...,i-1, is kept in a generalization rules set G (line 9) so that they will also be considered when building the new level i of the dynamic count tree.

The process we follow to find the generalization rules that will cause the least information loss in the data, is described in Algorithm 3. When a leaf node has a support lower than k, its siblings are the first to be considered for merging. This is because

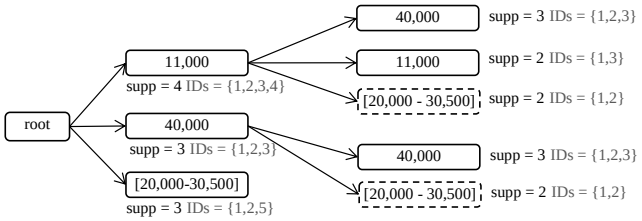

Fig. 3. Count tree T_2 for the dataset of Table1.

they share a common prefix (the path from the root to their common parent), which is an itemset of size i-1, and its support is ensured to be $\geq k$ at the previous step of ACD. Therefore only two values need to be generalized, the values of the leaves. The function $range(v_1, v_2)$ in line 11 returns the range between two values. If $v_1 < v_2$ then $range(v_1, v_2) = [v_1, v_2]$, else $range(v_1, v_2) = [v_2, v_1]$. If the combined support of the two paths is $\geq k$ then it is a candidate solution of this problematic itemset. For every candidate solution we measure the NCP that it would cause and choose the one that introduces the least distortion to the data. If the candidate solution with the lowest information loss gives $NCP < d$, we apply this generalization to the data. Otherwise, we parse the problematic path upwards to the root. At the next set of candidate generalizations we are looking for merges of both the leaves and their parent nodes, and so on, as shown in line 20 of Algorithm 3.

Note that in the worst-case scenario all values will be generalized to the maximum possible range, i.e., the data domain. Therefore, ACD will always find a k^m-anonymous solution to our problem.

Example 1. Consider the dataset of Table 1, let k=2, m=2. Figure 2 a) shows the count tree T_1, of height 1. Value 11,000 appears in records 1, 2, 3 and 4, so its support is 4, while 30,500 has support 1 as it appears only in record 2. Given k=2, this value must be generalized. The best generalization range is [20,000-30,500] as it affects less values in the dataset and thus gives lower NCP than the other options. This generalization is applied to both node 30,500 and 20,000 that fall in the chosen range. The two nodes are merged and their combined support is 3>k, as shown in Figure 2(b). In the next step, we add itemsets of size 2 to the count tree. T_2 is shown in Figure 3 where all leaves have supports at least k. The output of the algorithm is the k^m-anonymous Table 3.

4 Experimental Evaluation

We evaluated experimentally the performance of our algorithm on real datasets from the UCI repository [8]. The implementation was done in C++ and the experiments were performed on an Intel Core 2 Duo CPU at 2.53GHz with 4GB RAM, running Mac OS.

Algorithms. We compare our algorithm to Apriori algorithm (AA) algorithm from [1]. The AA is the state-of-the-art algorithm for creating k^m-anonymous datasets using generalization. It uses a fixed hierarchy and follows the a priori principle: first it creates a k^1-anonymous dataset, then a k^2-anonymous, up to k^m-anonymous. We had to slightly modify it to accommodate duplicate values in records. We also implemented AA in the same platform as our main algorithm ACD.

Data. We use the US Census Data 1990 Data Set [9] from UCI data mining repository. We selected 8 numerical attributes which refer to different types of income. We treated zeros as nulls and removed them from each record. The active domain ranges from 0-197297. The dataset contains approximately 2.5M records, but after eliminating the records that have zero values in all the selected attributes, we are left with approximately 1M records. The average record size was 2.27.

Parameters. We study the behavior of our algorithm with respect to the following parameters: a) k parameter of anonymity, b) the limit on attacker's knowledge m c)

Algorithm 3 Finds a Generalization that fixes a rare itemset **findGeneralizations**

Require: T_i {Count Tree}, f {leaf of a vulnerable itemset path}, k {privacy parameter}, d {NCP threshold}
Ensure: generalized path of f will have a support $\geq k$.

```
1: n = f
2: S = ∅
3: G_f = ∅ {Generalization rules}
4: for every s_j sibling of node n do
5:    S = S ∪ {s_j} {merge candidates}
6: for every node s_j ∈ S do
7:    if the combined support of s_j and n is ≥ k then
8:       NCP_j = NCP({v_n, v_{s_j} → range(v_n, v_{s_j})})
9:       if n is not a leaf then
10:         for every node nc in the path from n to leaf f do
11:            NCP_j = NCP_j + NCP({v_nc, v_{sc_j} → range(v_nc, v_{sc_j})}) {node sc_j is
               descendant of s_j, and it is at the same level as nc.}
12: find s_j ∈ S such that NCP_j is minimum
13: if NCP_j < d then
14:    g = range(v_n, v_{s_j})
15:    G_f = G_f ∪ g
16:    for every node nc in the path from n to leaf f do
17:       g = range(v_nc, v_{sc_j}) {sc_j is descendant of s_j, and at the same level as nc.}
18:       G_f = G_f ∪ g
19: else
20:    let node n be f's parent
21:    goto 2
22: return G_f
```

NCP threshold d, and d) the dataset size $|D|$. In every experiment we vary one of these parameters keeping others fixed. The default setting of our parameters is k = 10, m = 2, d = 0.001 and $|D|$ = 100000. To provide a fair comparison with AA we created a very detailed hierarchy which splits the active domain of 0-197297 to ranges of 100 and then creates a hierarchy with fanout of 2 that is used by AA.

Evaluation Metrics. We evaluate our method with respect to the execution time of our algorithm in seconds and the information loss in terms of NCP.

Anonymization quality In Figure 4 we depict the performance of the algorithms in terms of information loss. As k increases, the NCP in both algorithms increases sublinearly, but ACD causes a loss equal to half to 1/3 of that of AA. As the maximum size of the attacker's knowledge m increases, NCP increases superlinearly for both algorithms. However, for ACD it scales a lot better; the cost of AA rises to triple of that of ACD as m grows.

The behavior of NCP threshold as d changes is shown in the first graph of Figure 5. AA is not affected by d, thus we only depict the NCP of AA for the standard parameter setting (k=10, m=2) for reference. When d is small, ACD offers significantly more utility to the released data. Even when d is close to 1 (i.e., the maximum NCP value) our algorithm produces similar information loss as AA.

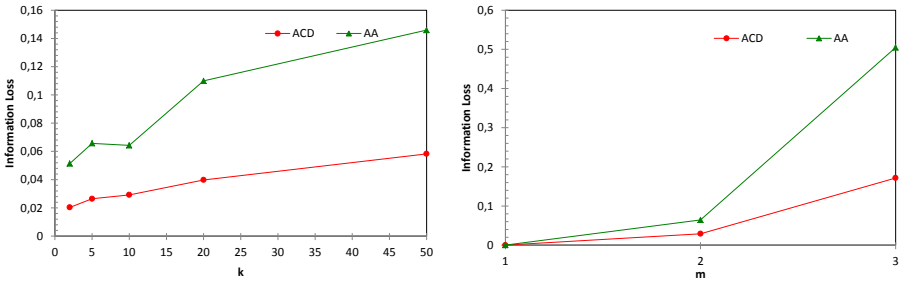

Fig. 4. Information Loss vs. k and m.

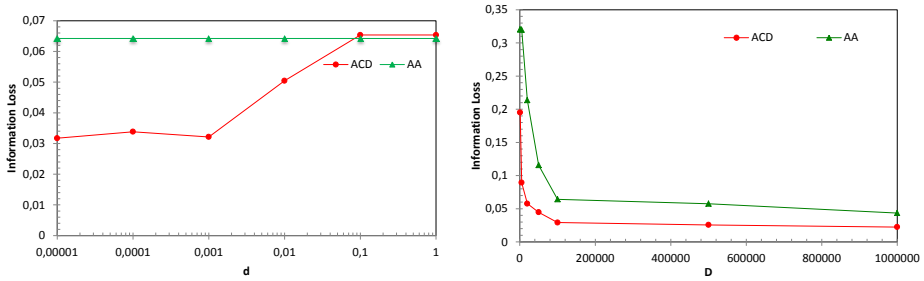

Fig. 5. Information Loss vs. d and $|D|$.

In the next graph of Figure 5, we vary the dataset size $|D|$. To perform this experiment we created seven random samples of our dataset of sizes 500,000, 100,000, 50,000, 25,000, 10,000, 5,000, 1,000 records. Each was randomly sampled from the previous one. Information loss of both algorithms decreases with the dataset size, with ACD outperforming AA in every dataset.

Execution Time. Figures 6 and 7 demonstrate the computational cost of our algorithm. Execution time is larger for small k values, and decreases monotonically as k increases. AA is faster than ACD, however the time difference is limited (around 25%) and insensitive to k.

Execution time grows sublinearly with reference to m for both algorithms, as for lager m, more itemsets of bigger sizes need to be considered and more levels of the count trees are needed.

In the next graph we depict the impact of d to running time. While ACD is slow for very small d, it approximates and slightly outperforms AA for $d = 0.0001$ and larger.

Finally, the scalability of our algorithm is shown in the second graph of Figure 7. The curve grows linearly with the dataset size $|D|$ for both algorithms.

In summary, ACD manages to greatly reduce the information loss, with the NCP of datasets anonymized with ACD being half or one third of the those that are anonymized with AA in most settings. This comes at the cost of increased CPU cost, but the overhead is limited to around 20%-40% in most cases.

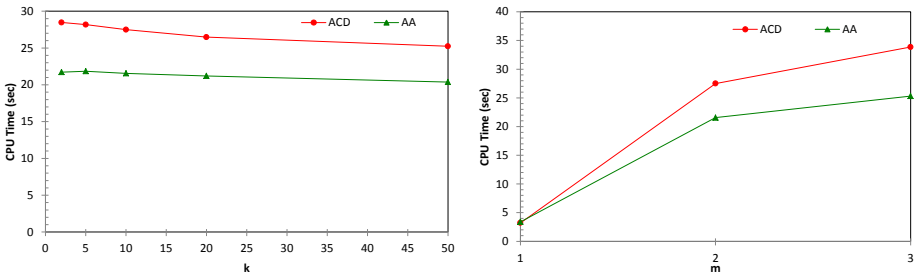

Fig. 6. Execution Time vs. k and m.

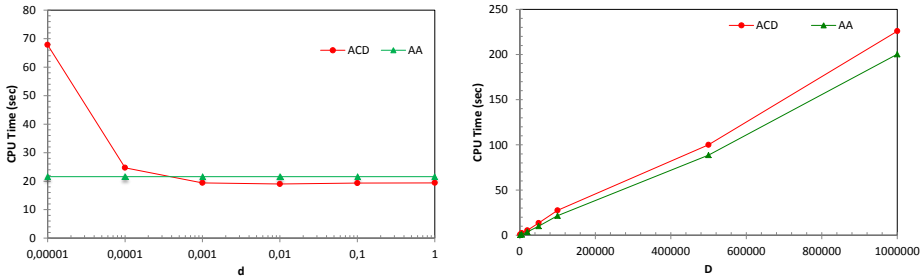

Fig. 7. Execution Time vs. d and $|D|$.

5 Related Work

The k-anonymity guarantee [2, 10, 11] was first proposed to protect individuals from identity disclosure, by demanding that each record in a published dataset should be indistinguishable from at least $k-1$ others, with respect to the quasi-identifiers. Most k-anonymization algorithms transform the data through generalization and suppression [12–21]. Other methods have also been proposed, such as permutation [22], perturbation [23, 24], microaggregation [25–27] and bucketization [28, 29, 4]. In general, k-anonymity is applied in Privacy-Preserving Data Publishing (PPDP) and Privacy-Preserving Data Mining (PPDM) scenarios. Nevertheless, it can be applied in other domains such as Privacy-Preserving Collaborative Filtering (PPCF) [30, 31].

A major difference between k-anonymity and our approach is the distinction between sensitive and non-sensitive values, as well as the assumption that the full set of QI is known. In our setting the problem is different, since any combination of m items can be used by an adversary as QIs. Our proposal extends the k^m-anonymity [1] and performs generalization without hierarchy on numerical attributes. Mondrian [17] also generalizes numerical attributes without the use of a hierarchy, however it applies classic k-anonymity, thus introducing more information loss to the released data.

k-anonymity was proven to be insufficient in preventing attribute disclosure. The ℓ-diversity guarantee proposed by Machanavajjhala et al. [32] demands that each EC have at least ℓ "well-represented" sensitive attribute (SA) values. The ℓ^+-diversity [33] extension sets a different privacy threshold to each SA value in order to reduce infor-

mation loss. Li et al. [34] proposed t-closeness which requires the distance between a sensitive attribute distribution in an EC and the global distribution of that attribute to be no greater than a threshold t. However, t-closeness lacks the flexibility of specifying different protection levels for different sensitive values and uses the Earth Mover Distance metric that is not suitable for measuring relative loss on individual sensitive attributes. To address these issues Cao and Karras proposed β-likeness [35].

These extensions of k-anonymity have a negative impact on the utility of the released data, as they introduce significant distortion. Relaxations of k-anonymity have been proposed [1, 36–40] aiming to provide a better trade-off between privacy and data utility. The pioneering work of Ghinita et al. [41] for sparse multidimensional data proposed a permutation method which first performs a grouping on transactions and then associates each group to a set of diversified sensitive values. In these data scenarios, it is very unlikely that the adversary has background knowledge of all QIs of his target [42]. Xu et al. [43] assume that the adversary has a limited knowledge of at most p non-sensitive attributes, while performing suppression on items that cause privacy leaks, but they still make the limiting distinction between sensitive and non-sensitive attributes.

Acknowledgements

This work was supported by the MEDA project within GSRT's KRIPIS action, funded by Greece and the European Regional Development Fund of the European Union under the O.P. Competitiveness and Entrepreneurship, NSRF 2007-2013. Olga Gkountouna's research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

6 Conclusions

In this work we studied the problem of k^m-anonymizing continuous data without the use of pre-defined data generalization hierarchies. We proposed `ACD`, a utility-preserving global-recoding heuristic algorithm. It greedily selects the best generalization ranges at each step, ensuring all itemsets of a particular size, at most m, appear at least k times in the dataset, thus satisfying the k^m-anonymity guarantee. We evaluated our method using real world datasets and compared our algorithm to `AA` [1] which uses generalization hierarchies for k^m-anonymization. Results show `ACD` preserves significantly more utility, at a small additional computational cost.

As future work, we plan to extend our solution to more complex attack models which will include both a partial and an aggregate knowledge [44] on the data values. We also wish to study k^m-anonymity under different data models.

References

1. M. Terrovitis, N. Mamoulis, and P. Kalnis, "Privacy-preserving Anonymization of Set-valued Data," *PVLDB*, vol. 1, no. 1, 2008.

2. L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *IJUFKS*, vol. 10, no. 5, 2002.
3. M. Terrovitis, N. Mamoulis, and P. Kalnis, "Local and global recoding methods for anonymizing set-valued data," *The VLDB Journal*, vol. 20, no. 1, pp. 83–106, 2011.
4. M. Terrovitis, N. Mamoulis, J. Liagouris, and S. Skiadopoulos, "Privacy preservation by disassociation," *Proceedings of the VLDB Endowment*, vol. 5, no. 10, pp. 944–955, 2012.
5. A. Meyerson and R. Williams, "On the Complexity of Optimal K-anonymity," in *PODS*, 2004, pp. 223–228.
6. J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *SIGMOD*, 2000, pp. 1–12.
7. J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu, "Utility-Based Anonymization Using Local Recoding," in *KDD*, 2006, pp. 785–790.
8. "Uci repository," http://archive.ics.uci.edu/ml/datasets.html.
9. "Uci repository us census data 1990 data set," http://archive.ics.uci.edu/ml/datasets/US+Census+Data+%281990%29.
10. P. Samarati and L. Sweeney, "Generalizing Data to Provide Anonymity when Disclosing Information (abstract)," in *PODS (see also Technical Report SRI-CSL-98-04)*, 1998.
11. P. Samarati, "Protecting respondents identities in microdata release," *TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.
12. L. Sweeney, "Datafly: A system for providing anonymity in medical data," in *Proc. of the International Conference on Database Security*, 1998, pp. 356–381.
13. V. S. Iyengar, "Transforming data to satisfy privacy constraints," in *SIGKDD*. ACM, 2002, pp. 279–288.
14. L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.
15. K. Wang, P. S. Yu, and S. Chakraborty, "Bottom-up generalization: A data mining solution to privacy protection," in *ICDM*. IEEE, 2004, pp. 249–256.
16. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *SIGMOD*. ACM, 2005, pp. 49–60.
17. K. LeFevre, D.-J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *ICDE*. IEEE, 2006, pp. 25–25.
18. B. C. Fung, K. Wang, and P. S. Yu, "Top-down specialization for information and privacy preservation," in *ICDE*. IEEE, 2005, pp. 205–216.
19. R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *ICDE*. IEEE, 2005, pp. 217–228.
20. K. El Emam, F. K. Dankar, R. Issa, E. Jonker, D. Amyot, E. Cogo, J.-P. Corriveau, M. Walker, S. Chowdhury, R. Vaillancourt *et al.*, "A globally optimal k-anonymity method for the de-identification of health data," *Journal of the American Medical Informatics Association*, vol. 16, no. 5, pp. 670–682, 2009.
21. F. Kohlmayer, F. Prasser, C. Eckert, A. Kemper, and K. A. Kuhn, "Flash: efficient, stable and optimal k-anonymity," in *PASSAT, SocialCom*. IEEE, 2012, pp. 708–717.
22. Q. Zhang, N. Koudas, D. Srivastava, and T. Yu, "Aggregate query answering on anonymized tables," in *ICDE*. IEEE, 2007, pp. 116–125.
23. A. Evfimievski, J. Gehrke, and R. Srikant, "Limiting privacy breaches in privacy preserving data mining," in *ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2003, pp. 211–222.
24. V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," *TKDE*, vol. 16, no. 4, pp. 434–447, 2004.
25. J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogeneous k-anonymity through microaggregation," *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 195–212, 2005.

26. J. Domingo-Ferrer, A. Solanas, and A. Martinez-Balleste, "Privacy in statistical databases: k-anonymity through microaggregation." in *GrC*, 2006, pp. 774–777.
27. J. Domingo-Ferrer, "Microaggregation: achieving k-anonymity with quasi-optimal data quality," in *European Conference on Quality in Survey Statistics*, 2006.
28. X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *VLDB*. VLDB Endowment, 2006, pp. 139–150.
29. T. Li, N. Li, J. Zhang, and I. Molloy, "Slicing: A new approach for privacy preserving data publishing," *TKDE*, vol. 24, no. 3, pp. 561–574, 2012.
30. F. Casino, C. Patsakis, D. Puig, and A. Solanas, "On privacy preserving collaborative filtering: Current trends, open problems, and new issues," in *e-Business Engineering (ICEBE)*. IEEE, 2013, pp. 244–249.
31. F. Casino, J. Domingo-Ferrer, C. Patsakis, D. Puig, and A. Solanas, "Privacy preserving collaborative filtering with k-anonymity through microaggregation," in *e-Business Engineering (ICEBE)*. IEEE, 2013, pp. 490–497.
32. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *TKDD*, vol. 1, no. 1, p. 3, 2007.
33. J. Liu and K. Wang, "On optimal anonymization for l^+-diversity," in *ICDE*. IEEE, 2010, pp. 213–224.
34. N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *ICDE*. IEEE, 2007, pp. 106–115.
35. J. Cao and P. Karras, "Publishing microdata with a robust privacy guarantee," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1388–1399, 2012.
36. K. Wang and B. Fung, "Anonymizing sequential releases," in *SIGKDD*. ACM, 2006, pp. 414–423.
37. A. Gionis, A. Mazza, and T. Tassa, "k-anonymization revisited," in *ICDE*. IEEE, 2008, pp. 744–753.
38. W. K. Wong, N. Mamoulis, and D. W. L. Cheung, "Non-homogeneous generalization in privacy preserving data publishing," in *SIGMOD*. ACM, 2010, pp. 747–758.
39. T. Tassa, A. Mazza, and A. Gionis, "k-concealment: An alternative model of k-type anonymity." *Transactions on Data Privacy*, vol. 5, no. 1, pp. 189–222, 2012.
40. K. Stokes and V. Torra, "n-confusion: a generalization of k-anonymity," in *EDBT/ICDT Workshops*. ACM, 2012, pp. 211–215.
41. G. Ghinita, Y. Tao, and P. Kalnis, "On the Anonymization of Sparse High-Dimensional Data," in *ICDE*, 2008.
42. A. Zigomitros, A. Solanas, and C. Patsakis, "The role of inference in the anonymization of medical records," in *Computer-Based Medical Systems (CBMS)*, 2014.
43. Y. Xu, K. Wang, A. W.-C. Fu, and P. S. Yu, "Anonymizing transaction databases for publication," in *ACM SIGKDD*. ACM, 2008, pp. 767–775.
44. O. Gkountouna, K. Lepenioti, and M. Terrovitis, "Privacy against aggregate knowledge attacks," in *PrivDB, Data Engineering Workshops (ICDEW)*. IEEE, 2013, pp. 99–103.