Aggregate Query Answering on Anonymized Tables

Qing Zhang[a] Nick Koudas[b] Divesh Srivastava[c] Ting Yu[a]
[a]North Carolina State University [b]University of Toronto [c]AT&T Labs - Research
{qzhang4, tyu}@ncsu.edu koudas@cs.toronto.edu divesh@research.att.com

Abstract

Privacy is a serious concern when microdata need to be released for ad hoc analyses. The privacy goals of existing privacy protection approaches (e.g., k-anonymity and ℓ-diversity) are suitable only for categorical sensitive attributes. Since applying them directly to numerical sensitive attributes (e.g., salary) may result in undesirable information leakage, we propose privacy goals to better capture the need of privacy protection for numerical sensitive attributes. Complementing the desire for privacy is the need to support ad hoc aggregate analyses over microdata. Existing generalization-based anonymization approaches cannot answer aggregate queries with reasonable accuracy. We present a general framework of permutation-based anonymization to support accurate answering of aggregate queries and show that, for the same grouping, permutation-based techniques can always answer aggregate queries more accurately than generalization-based approaches. We further propose several criteria to optimize permutations for accurate answering of aggregate queries, and develop efficient algorithms for each criterion.

1 Introduction

Compared with data dissemination in pre-aggregated or statistical forms, the release of microdata offers significant advantages in terms of information availability, which make it particularly suitable for ad hoc analyses in a variety of domains such as public health and population studies. However, the release of microdata raises privacy concerns of revealing private information of individuals. Simple de-identification (i.e., removing explicit identifiers, e.g., name, SSN) has been shown to be insufficient to protect an individual's privacy. One's other attributes (so-called *quasi-identifiers*, such as age, zip code and date of birth) are usually needed for data analyses, and thus are kept after de-identification. This often allows individuals' sensitive information to be revealed when microdata are linked with publicly available information through quasi-identifiers.

k-anonymization and ℓ-diversity are techniques to address this privacy problem. Through domain generalization, they guarantee that even if publicly available information is linked with records in a microdata database, a sensitive attribute value can at most be related to a group of no less than k entities (and having no less than ℓ distinct sensitive attribute values), instead of to a specific individual.

The privacy goal of k-anonymization and ℓ-diversity is suitable for categorical sensitive attributes, such as the disease attribute in a patient record table, where different attribute values are incomparable. In practice, besides categorical attributes, many sensitive attributes in microdata databases are numerical, e.g., salary, investment gains or losses, credit score, white blood cell count, etc. Applying existing privacy goals of k-anonymity and ℓ-diversity is often not sufficient to protect numerical attributes. For example, even when ℓ-diversity is satisfied, if the salary values fall in a narrow range, sensitive information is revealed. Thus, it is important to define new privacy goals for the protection of numerical sensitive attributes in microdata.

Complementing the desire for privacy of microdata is the need to support ad hoc aggregate analyses that select subsets of records based on arbitrary conditions on the quasi-identifiers and compute aggregates over sensitive attributes (e.g., what is the average salary of men over age 50 in Texas?). Since most existing approaches achieve privacy through generalization of quasi-identifiers, they cannot answer aggregate queries with any reasonable accuracy, thereby reducing the utility of the released microdata. In this paper, we investigate effective techniques to support accurate aggregate query answering on microdata while preserving privacy.

The essential goal of privacy protection is to break the association between the identifiers in publicly available information and the sensitive attributes in microdata. We observe that there are multiple ways to do so. Existing generalization-based approaches weaken the link between quasi-identifiers in publicly available databases and microdata. We propose *permutation*-based approaches to reduce the association between quasi-identifiers and sensitive attributes, and make the following contributions.

	ID	Quasi-identifiers			Sensitive
tuple ID	name	age	zipcode	gender	salary
1	Alex	35	27101	M	$54,000
2	Bob	38	27120	M	$55,000
3	Carl	40	27130	M	$56,000
4	Debra	41	27229	F	$65,000
5	Elain	43	27269	F	$75,000
6	Frank	47	27243	M	$70,000
7	Gary	52	27656	M	$80,000
8	Helen	53	27686	F	$75,000
9	Jason	58	27635	M	$85,000

Figure 1. An example microdata table

First, we propose a new privacy goal to better capture privacy protection for numerical sensitive attributes (section 3). Besides requiring a group of sensitive attribute values to have no less than k distinct values, the proposed privacy goal further requires the range of the group to be larger than a certain threshold e; such a threshold prevents an attacker from determining an individual's sensitive attribute value in a narrow range.

Second, we propose a general framework of permutation-based techniques to support accurate answering of aggregate queries, while protecting privacy (section 4). Given the same grouping of tuples, permutation-based anonymization techniques can always answer aggregate queries more accurately than generalization-based approaches.

Third, we design query-rewriting algorithms and auxiliary relations to ensure that existing DBMSs can be used directly to support aggregate query answering in permuted microdata (section 5). The auxiliary relations are completely derived from an anonymized microdata database, and thus do not compromise the privacy of microdata.

Fourth, we identify several alternative criteria to optimize permutations for accurate answering of aggregate queries, and develop efficient algorithms for each criterion (section 6). Since the permutation-based approach is not constrained by domain generalization hierarchies, our optimization algorithms obtain partitions of microdata that can help answer aggregate queries very accurately.

Finally, we conduct comprehensive experiments on both real and synthetic data sets to demonstrate the advantages of the proposed techniques (section 7).

We next present a detailed example illustrating our approach and its benefits.

2 Examples

Consider the population table shown in figure 1, which needs to be shared for business and economic research to answer, e.g., the following aggregate queries:

Query 1. The sum of salaries of those with age in [35,55].

Query 2. The minimum salary of females.

		Quasi-identifiers			Sensitive
group ID	tuple ID	age	zipcode	gender	salary
1	1	[31-40]	271*	*	$56,000
1	2	[31-40]	271*	*	$54,000
1	3	[31-40]	271*	*	$55,000
2	4	[41-50]	272*	*	$65,000
2	5	[41-50]	272*	*	$75,000
2	6	[41-50]	272*	*	$70,000
3	7	[51-60]	276*	*	$80,000
3	8	[51-60]	276*	*	$75,000
3	9	[51-60]	276*	*	$85,000

Figure 2. An example 3-anonymity microdata table after generalization. This table also satisfies 3-diversity

Even when the population table is de-identified (identity attribute "name" is removed), the quasi-identifier attributes "age", "zip code" and "gender" can often be linked with publicly available information containing both the identity and quasi-identifier attributes, permitting sensitive attributes such as "salary" of individuals to be revealed.

Current approaches to addressing this problem generalize the domains of quasi-identifiers so that many tuples will have the same quasi-identifiers. Figure 2 shows such a generalization, where "age" is generalized to a range of width 10, "zip code" only keeps the first 3 digits, and "gender" is totally suppressed. The resulting table satisfies 3-anonymity, which means that after generalization each tuple can find at least two other tuples with the same quasi-identifiers. It also satisfies 3-diversity since among those tuples with the same quasi-identifiers there are at least 3 different sensitive attribute values. A generalized table in fact forms a partition of the original microdata table, where each group contains tuples with the same generalized quasi-identifiers. Figure 2 also shows the group ID of each tuple.

In this paper, we propose a permutation-based approach to anonymization. In our approach, tuples in the table are partitioned into several groups such that each group has at least k different sensitive attribute values. We then perform a permutation between the tuples' quasi-identifiers with their sensitive attribute inside each group. Figure 3 shows the table of figure 1 after permutation, where each group has 3 different sensitive attribute values. In fact, the partition used in figure 3 is the same as the one from the generalization in figure 2. In section 4, we will show that the permutation-based approach will achieve the same privacy protection as existing generalization-based approaches. A significant benefit of our permutation-based approach is that it will provide more accurate answers to aggregate queries.

Consider the two queries previously mentioned in this section. Since both approaches introduce imprecise information, they cannot always get the correct answer for aggregate queries. Instead, for each approach, we can get de-

		Quasi-identifiers			Sensitive
group ID	tuple ID	age	zipcode	gender	salary
1	1	40	27130	M	$54,000
1	2	38	27120	M	$55,000
1	3	35	27101	M	$56,000
2	4	41	27229	F	$65,000
2	5	43	27269	F	$70,000
2	6	47	27243	M	$75,000
3	7	52	27656	M	$75,000
3	8	53	27686	F	$80,000
3	9	58	27635	M	$85,000

Figure 3. An example 3-anonymous microdata table after permutation

terministic lower and upper bounds of the correct answer. We compare the accuracy of the bounds of each approach.

Query 1: The sum of salaries of those with age between 35 and 55. In both tables, all the tuples in the second group should be included in the aggregation. But, by using the generalized table, we face a difficulty when dealing with tuples in the first and the third groups. Since only generalized ranges [31-40] and [51-60] of ages are available, to get a correct lower/upper bound, we have to assume that none/all of the tuples participate in the aggregation. Hence, the bounds using figure 2 can only be [$210K, $615K].

On the other hand, in our permutation-based approach, we know exactly how many tuples are included in the aggregation from each group, which helps to get more accurate bounds. For this query, based on figure 3, we know that 3 and 2 tuples in groups 1 and 3 respectively participate in the aggregation. Therefore, the bounds will be [$530K, $540K], which are much more accurate than those derived from figure 2.

Query 2: The minimum salary of females. It is very difficult to answer this query using figure 2. The "gender" attribute is totally suppressed in order to achieve 3-anonymity. We do not even know whether a tuple exists with gender female in the original table. Even if we assume that there is at least one female in the table, the best bound we can get is [$54K, $85K].

From the permuted table, we know that there is a female in each of groups 2 and 3, but none in group 1. Since all the salaries in group 3 are higher than those in group 2, we can conclude that the minimum salary of females in this table is between [$65K, $75K]. Again, this is more accurate than the bounds derived from figure 2. In section 4, we will show that, given the same partitions, the permutation-based approach always produces more accurate bounds for aggregation queries than the generalization-based approach.

One nice property is that the lower and upper bounds for an aggregation operation over the whole table can be computed efficiently by combining bounds over each group of the partition. To facilitate efficient query answering over a permuted table, we propose to use a *help table*, which precomputes the bounds for each group and all the possible "numbers of hits". Given an aggregate query, we can simply rewrite it to query the permuted table, determining the number of hits of each group, and then join this information with the help table to quickly get the bounds for the whole query result. An example help table, for the table of figure 1, is shown in figure 4. Due to space reasons, we show only the bounds for SUM and MIN in the table. It can include those for other aggregation operations as well.

group ID	hits	sum-l-b	sum-u-b	min-l-b	min-u-b
1	1	$54K	$56K	$54K	$56K
1	2	$109K	$111K	$54K	$55K
1	3	$165K	$165K	$54K	$54K
2	1	$65K	$75k	$65K	$75K
2	2	$135K	$145K	$65K	$70K
2	3	$210K	$210K	$65K	$65K
3	1	$75K	$85K	$75K	$85K
3	2	$155K	$165K	$75K	$80K
3	3	$240K	$240K	$75K	$75K

Figure 4. An example help table

Both the generalized and the permuted tables satisfy 3-diversity. However, their partitions may have a problem in terms of privacy. If an attacker knows that the age of Alex is 35 and his zipcode is 27101, he is able to derive that Alex's salary is between $54K and $56K. Though the attacker cannot know the exact salary of Alex, this range might be narrow enough to be sensitive. This example shows that for numerical sensitive attributes, besides the number of distinct values in each group, we also need to consider the range of values of each group to prevent the above inference. For this reason, in this paper we introduce a privacy parameter e, and further require the range of the k distinct values in a group to be no less than e. We call this privacy objective (k, e)-anonymity. For instance, the above generalized and permuted tables only satisfy $(3, 2000)$-anonymity but violate $(3, 10000)$-anonymity.

In the following sections, we formally describe (k, e)-anonymity and a permutation-based approach to anonymization of microdata.

3 Privacy Goal

Microdata. There are three types of attributes in an *original* microdata table $\mathcal{M}$: identifiers (ID), quasi-identifiers ($\{QI_1, \ldots QI_k\}$) and sensitive attributes. For simplicity, we assume there is only one sensitive attribute S in a microdata table, and focus on numeric sensitive attributes; our techniques can be easily adapted to process categorical sensitive attributes as well. Figure 1 is an example of an original microdata table where the identifier, quasi-identifiers, and sensitive attribute are shown. A de-identified microdata table $\mathcal{D}$ is a projection of $\mathcal{M}$ over quasi-identifiers and sensitive attributes. We call the projection of $\mathcal{M}$ over ID and S the sensitive information table, denoted $\mathcal{S}$.

Public information. We model publicly available information as a table $\mathcal{P}$ with the following attributes $\{ID, QI_1, \ldots, QI_k\}$. In practice, there may exist multiple sources of public information, such as county real estate databases and voter registration records. The above model represents the overall public information that an attacker may derive when combining information from multiple sources.

Aggregate queries. We consider queries that select subsets of records from a microdata table based on arbitrary conditions on the quasi-identifiers and compute aggregates over the sensitive attribute. Such aggregate queries are important during microdata analysis in a variety of domains. Since the domain of the sensitive attribute is assumed to be numeric, many SQL aggregation operations, such as COUNT, SUM, AVERAGE, MIN and MAX, can be used in aggregate queries.

Privacy goal. Privacy is violated when an attacker successfully recovers one or more tuples in the sensitive information table $\mathcal{S}$. Formally, we have the following privacy definition based on the one by Yao et al. [14].

Definition 3.1. *Each tuple on (ID, S) is called an* association. *A set $\mathcal{A}$ of associations on (ID, S) is called an* association cover *if all the tuples in A have the same ID value and $\mathcal{A} \cap \mathcal{S} \neq \emptyset$. An association cover of size k is called a k*-association-cover.

For example, consider the microdata in figure 1, $\{(Alex, \$54,000), (Alex, \$55,000), (Alex, \$56,000)\}$ is a 3-association cover.

Definition 3.2. *A de-identified microdata database $\mathcal{D}$ satisfies k*-anonymity *if from $\mathcal{D}$ and any given public database $\mathcal{P}$, an attacker cannot derive any association cover with size less than k.*

The above definition captures the essence of privacy in microdata, i.e., preventing the association between an individual's ID and its sensitive attribute value. The originally proposed concept of k-anonymity was defined specifically for generalization based approaches. It required that, after generalization, for each tuple t in the table, there should exist no less than $k-1$ other tuples having quasi-identifiers equal to those of t. This original definition can be viewed as the goal for generalization in order to achieve k-anonymity. Definition 3.2, on the other hand, is declarative and independent of specific techniques for anonymization. Therefore, it serves as a good privacy definition for the comparison of different anonymization techniques.

As shown in section 2, for numeric-valued attributes, preventing attackers from deriving an association cover of size less than k may not be enough to protect one's privacy, especially when the range of attribute values in the association cover is small. Therefore, we propose the following extended definition for the protection of numeric-valued sensitive attributes:

Definition 3.3. *A de-identified microdata database $\mathcal{D}$ satisfies (k, e)*-anonymity *if given $\mathcal{D}$ and any given public database $\mathcal{P}$, any association cover that an attacker can derive satisfies: (1) the size of the association cover is no less k; and (2) the range of the sensitive attribute values in the association cover is no less than e.*

4 Anonymization through Permutation

The essential reason that an attacker may recover an individual's sensitive attribute value is the existence of the following three links: (1) the link between the identifier and quasi-identifiers in the public database $\mathcal{P}$; (2) the link between the quasi-identifiers in $\mathcal{P}$ and those in the de-identified microdata $\mathcal{D}$; and (3) the link between quasi-identifiers and the sensitive attribute in $\mathcal{D}$. Breaking or weakening the associations of any of the above links will help protect privacy. Domain generalization actually weakens the second and the third links.

In this paper, we propose to break *only* the third link through permutation. Given a set of tuples in a de-identified microdata table, we randomly permute the association between quasi-identifiers and the sensitive attribute instead of using domain generalization on the quasi-identifiers. Intuitively, even if an attacker can link an individual's identifier with a tuple's quasi-identifier (for example through background knowledge), he will not be able to know with certainty the exact value of the individual's sensitive attribute.

Definition 4.1. *Let $T = \{t_1, \ldots, t_n\}$ be a table with attributes $\{a_1, \ldots, a_m\}$, and p be a random permutation over $\{1, \ldots, n\}$. We define the permutation of T, denoted $p(T, \{a_1, \ldots, a_l\}, \{a_{l+1}, \ldots, a_m\})$ as the set of tuples $\{t'_i \mid \forall j, 1 \leq j \leq l, t'_i[a_j] = t_i[a_j]$ and $\forall j, l+1 \leq j \leq m, t'_i[a_j] = t_{p(i)}[a_j]\}$.*

Definition 4.2. *Let $\mathcal{D}$ be a de-identified microdata table with attributes $\{QI_1, \ldots, QI_k, S\}$, and $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a partition of $\mathcal{D}$. A group $\mathcal{D}_i$ is (k, e)*-anonymous *if the projection of $\mathcal{D}_i$ over the sensitive attribute S contains at least k different values, and the range of these different values in $\mathcal{D}_i$ is at least e. We say the partition is (k, e)*-anonymous *if every $\mathcal{D}_i$ in the partition is (k, e)-anonymous. We denote $\mathcal{D}'_i = p(\mathcal{D}_i, \{QI_1, \ldots, QI_k\}, \{S\})$. $\mathcal{D}_p = \bigcup_{i=1,\ldots,n} \mathcal{D}'_i$ is a (k, e)-anonymous permutation of $\mathcal{D}$.*

As an example, figure 3 shows a (3,2000)-anonymous permutation of the table in figure 1. Note that, when publishing the permuted table, we also attach to each tuple the

ID of the group that it belongs to. In other words, compared with the schema of the de-identified table, the schema of the anonymized table also has an extra attribute "group ID".

Theorem 1. *Given a (k, e)-anonymous permutation $\mathcal{D}_p$ and a public database $\mathcal{P}$, any association cover that an attacker can derive satisfies (k, e)-anonymity.* □

5 Aggregate Query Answering

Given a (k, e)-anonymous permutation $\mathcal{D}_p$ and an arbitrary query condition over quasi-identifiers, since the quasi-identifiers of a tuple are unchanged, we know exactly how many tuples satisfy the condition in each group $\mathcal{D}_i$ of the partition. Suppose this number is m_i. Due to the random permutation between quasi-identifiers and the sensitive attribute in $\mathcal{D}_i$, the actual result of the aggregate in $\mathcal{D}_i$ may be over the sensitive attribute of any m_i tuples in the group. Thus, in the worst case, there may be totally $C(|\mathcal{D}_i|, m_i)$ different results for the aggregate. It would be too expensive to enumerate all the possible results for large groups. Instead, we are interested in efficiently computing statistics such as the lower and upper bounds of all the possible aggregates, which will be very useful for ad hoc analyses.

5.1 Lower and Upper Bounds

Let m_i be the number of tuples in $\mathcal{D}_i$ that satisfy the condition of an aggregate query Q. If the aggregation operation is monotonic, then the lower/upper bound of the result of Q in $\mathcal{D}_i$ is the aggregation of the m_i smallest/largest sensitive attribute values in $\mathcal{D}_i$. Since standard SQL aggregation functions, such as SUM, MIN, MAX and COUNT, are monotonic, the lower/upper bounds of the result of Q in each $\mathcal{D}_i$ can be efficiently computed.

The lower/upper bounds of the query over the whole microdata table can be derived from those of each $\mathcal{D}_i$, depending on the aggregation function. For SUM, the overall lower/upper bound is the summation of the lower/upper bound of each $\mathcal{D}_i$. Since we know the total number of tuples satisfying the query condition, the lower/upper bounds of AVERAGE can be directly computed from those of SUM. It is also easy to see that the overall lower/upper bounds of MIN is the minimum among the lower/upper bounds of all the $\mathcal{D}_i$. The bounds for MAX can be obtained similarly.

Theorem 2. *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_r\}$ be a partition of $\mathcal{D}$ defined by a generalization. Given any aggregate query, the lower and upper bounds given by the generalized table always include those given by the permuted table using the same partition.* □

5.2 Auxiliary Relation and Query Rewriting

We observe that, for the same aggregation operation, no matter what the query condition is, as long as the number of tuples satisfying the condition in each $\mathcal{D}_i$ is the same, the bounds of the aggregation in $\mathcal{D}_i$ remain unchanged. Therefore, we do not need to compute the bounds for each $\mathcal{D}_i$ on the fly when answering a query. Instead, we propose to create a *help table* to facilitate efficient query answering.

The primary key of the help table is "group ID" and "hits", where the former identifies a group in the partition, and the latter represents the number of tuples in the group satisfying a query condition. For each group in the partition and the number of hits, the table lists the lower and upper bounds for each aggregate on the sensitive attribute. Figure 4 shows the help table for the permuted table in figure 3 for the aggregates SUM and MIN. The number of tuples in the help table is the same as in the microdata table.

Given an aggregate query of the form "select agg(sensitive-attribute) from permuted-table where C", we rewrite it to get the bounds of the query result. The rewritten query first counts the number of tuples of each group that satisfy condition C in the permuted table PT. It is then joined with the help table HT, and the bounds of the aggregation of each group are combined to compute the bounds for the final query result. This computation is easily expressed in SQL over the permuted and help tables.

We emphasize that the help table can be constructed directly from the permuted table without requiring any access to the original microdata table. Therefore, the use of the help table does not compromise the privacy of microdata in any way. Further, we only need to compute the help table once for a released microdata database. It can be done offline, without affecting the performance of ad hoc analyses.

6 Criteria for (k, e)-Anonymous Partition

We have shown that, given the same partition, the anonymized table obtained through permutation will always answer aggregate queries more accurately than that obtained through domain generalization. However, given an arbitrary partition, even with the permutation-based approach, it may not get satisfactory answers to aggregate queries. In this section, we turn our discussion to the problem of generating "good" (k, e)-anonymous partitions to produce accurate answers to aggregate queries.

Formally, let D denote a total order of the multiset of sensitive attribute values $D = x_1, x_2, \ldots, x_n$, and $P = \{G_1, \ldots, G_m\}$ be a partition of D. Since D is a totally ordered multiset, we denote the indices of the first and the last data point in a group G_i as min_i and max_i respectively. Thus, the range of G_i is obtained by $[x_{min_i}, x_{max_i}]$. Let $E(G_i)$ denote an error measure defined on a group D_i, and

F be a point-wise additive function. We can now formally define the optimal partition problem:

Problem 6.1 (Optimal Partition). *Given a total order D of a sensitive attribute, obtain a (k,e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ that minimizes $F(E(G_1), E(G_2), \ldots E(G_m))$ for suitable choices of $F()$ and $E()$.*

Given a point query x_q, if $x_q \in G_i$, our scheme will return any point inside G_i as an answer. As a result, the *maximum* error incurred for any point query inside a group G_i is $E(G_i) = x_{max_i} - x_{min_i}$. Therefore, intuitively, the smaller the range of each group, the smaller error will be introduced to the answer of aggregate queries. Since all the groups of P may be used for querying, we would like to define the function F in a way that the error across all groups, assuming a uniform random workload of point queries, is minimized. It is natural to aim to minimize the *additive* error or the *max* error across all groups. Thus candidate point-wise additive functions are sum or max. We call the optimization problems using the sum and the max functions the *minimum sum-of-error problem* and the *minimum max-of-error problem* respectively. We denote the sum and the max of errors of all the groups in a partition P as $sum\_of\_error(P)$ and $max\_of\_error(P)$ respectively.

6.1 The Minimum Sum-of-Error Problem

Without loss of generality, we assume that $G_1, \ldots, G_m$ are ordered according to the index of the minimum value of each group, i.e., $i < j$ implies that $min_i < min_j$, which also means that $x_{min_i} \leq x_{min_j}$.

Lemma 6.1. *There exists an optimal partition $P = \{G_1, \ldots, G_m\}$ to the minimum sum-of-error problem such that for any groups G_i and G_{i+1}, we have $max_i < min_{i+1}$.* □

Lemma 6.1 shows that the ranges of groups in an optimal partition are disjoint. It suggests that this problem has the optimal substructure property, thus it is amenable to dynamic programming solutions. Let $f(i)$ denote the minimum cost way to partition $x_1, \ldots, x_i$ into a number of groups, say m^*, such that the partition is (k, e)-anonymous for $x_1, \ldots, x_i$. Then

$$f(i) = min_{1 \leq d \leq i} F(f(d-1), E(\{x_d, \ldots, x_i\})) \quad (1)$$

Thus, the optimal solution for partitioning the data points $x_1, \ldots x_i$ into m^* groups is equal to the minimum cost way of extending (according to the point wise additive function F, which in this problem is sum) the optimal $m^* - 1$ partitioning of $x_1, \ldots x_{d-1}$ (for some d, $1 \leq d \leq i$) with the group $\{x_d, \ldots, x_i\}$. It is easy to see that this dynamic programming algorithm runs in $O(n^2)$, where n is the number of tuples in the table.

6.2 The Minimum Max-of-Error Problem

Though seemingly similar to the minimum sum-of-error problem, the minimum max-of-error problem turns out to be much more complex. We have shown that the groups of an optimal partition in the sum-of-error problem are disjoint, i.e., the ranges of different groups are not overlapping (except possibly the boundary where $x_{max_i} = x_{min_{i+1}}$). This property significantly reduces the search space for an optimal partition.

The non-overlapping property, however, does not hold for optimal partitions for the minimum max-of-error problem. As a simple example, consider the following set of sensitive attributes $\{1, 2, 3, 5, 5, 6, 6, 8\}$, the only optimal partition for $(4, 5)$-anonymity is $G_1 = \{1, 2, 5, 6\}$ and $G_2 = \{3, 5, 6, 8\}$, where G_1's range $(1, 6)$ overlaps with G_2's range $(3, 8)$.

On the other hand, we observe that the minimum max-of-error problem has the following property.

Lemma 6.2. *There exists an optimal (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ for D such that there are no more than two groups whose ranges overlap with each other. In other words, there is no value in the domain of the sensitive attribute such that the value falls into the ranges of more than two groups.* □

Let $P = \{G_1, \ldots, G_m\}$ be an optimal (k, e)-anonymous partition that satisfies the above property. Consider the first two groups G_1 and G_2. If the ranges of G_1 and G_2 do not overlap, then G_1 contains all the values from $x_{min_1} = x_1$ to x_{max_1} in D. Then the remaining groups in fact form an optimal partition for the rest of the items $x_{max_1+1}, \ldots, x_n$.

On the other hand, suppose the ranges of G_1 and G_2 overlap, which means $min_2 < max_1$. We divide G_2 into two parts, the former part G_{2f} which includes those items less than or equal to x_{max_1}, and the latter part G_{2l} which includes those items greater than x_{max_1}. Let t be the smallest index of items in D such that $x_t > x_{max_1}$. According to lemma 6.2, G_1 and G_{2f} combined together include all the values in D that are less than x_t. Further, we have $x_t \in G_{2l}$. Otherwise, suppose $x_t \in G_i, i > 2$. Then x_t must be the smallest value of G_i. This allows us to merge G_{2f} with G_1 and G_{2l} with G_i. The resulting partition P' is still optimal, and there is no overlap between the first two groups in P'.

Based on the above observation, we have that G_{2l}, G_3, $\ldots$, G_m forms a partition of $x_t, x_{t+1}, \ldots, x_n$. Except G_{2l}, every group is (k, e)-anonymous. For G_{2l}, it has at least $k - d$ distinct values where d is the number of distinct values in G_{2f}. Further, the width of the range of G_{2l} is no less than $e - r$, where $r = x_t - x_{min_2}$. In other words, given any partition $P' = \{G'_1, \ldots, G'_z\}$ of $x_t, \ldots, x_n$, such that G'_1 is $(k - d, e - r)$-anonymous and the rest are (k, e)-anonymous, if the maximum of $E(G'_1) + r$, $E(G'_2)$, $\ldots$, $E(G'_z)$ is minimized, then the partition $\{G_1, G_{2f} \cup G'_1, G'_2, \ldots, G'_z\}$ also forms an optimal (k, e)-anonymous partition of D.

Therefore, we study the following more general optimization problem:

Problem 6.2. *Given d and r, obtain a partition $P = \{G_1, \ldots, G_m\}$ of D, where G_1 is $(k-d, e-r)$-anonymous and the rest groups are (k, e)-anonymous, such that $max(E(G_1) + r, E(G_2), \ldots, E(G_m))$ is minimized.*

Clearly, the minimum max-of-error problem is a special case of the above problem where $d = 0$ and $r = 0$.

The above argument shows that the above problem has the optimal substructure property. Intuitively, for each possible max_1 and min_2, we move as many values between x_{min_2} and x_{max_1} as possible to G_{2f}, the first part of G_2, as long as G_1 still has $k - d$ distinct values. After this step, we denote the number of distinct values in G_{2f} as $\#(min_1, max_1, min_2, d)$, as it is determined by these four parameters. Let $g(d, r, i)$ denote the minimum cost way to partition $x_i, \ldots, x_n$ such that the first group is $(k - d, e - r)$-anonymous and the rest groups are (k, e)-anonymous. Then we have $g(d, r, i) = min_{i \leq u \leq n, i \leq v \leq u-1} max(g(d - \#(i, u, v, d), r - (x_{u+1} - x_v), u + 1), x_u - x_i + r)$. Details of the algorithm are in the full version of this paper [5].

The purpose of our discussion so far is to show that the minimum max-of-error problem is in fact tractable. However, with complexity of $O(n^6)$, it is far from practical. Instead, it is desirable to design efficient approximation algorithms for the problem. For this purpose, we limit our search space to those partitions whose groups do not overlap with each other, and consider the following problem:

Problem 6.3. *Obtain a (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ of D, where the ranges of any two groups in the partition do not overlap, such that $max(E(G_1), \ldots, E(G_m))$ is minimized.*

We call the above problem the non-overlapping minimum max-of-error problem. With a similar argument to that of the optimal solution to the minimum sum-of-error problem, it is not hard to see that this problem also has the optimal substructure property, and thus can be solved by dynamic programming with $O(n^2)$ in both space and time.

Theorem 3. *Let P and P' be the optimal partitions of D of the minimum max-of-error problem and the non-overlapping minimum max-of-error problem. Then $min\_max\_error(P') \leq 2 \cdot min\_max\_error(P)$.* □

7 Experiments

7.1 Data Sets and Experiment Design

Our experiments are conducted on the Adult Database from the UCI Machine Learning Repository [12]. The database is obtained from the US Census data, and contains 14 attributes and over 48,000 tuples. The same database has been used in previous works on k-anonymity and ℓ-diversity [6, 8]. We choose the same quasi-identifiers (which contain 8 attributes) as that used in previous works. Since our approach focuses on numerical-valued sensitive attributes, in the experiments we choose "capital loss" as the sensitive attribute. In particular, we are interested in those people who do have capital loss. Therefore, we remove those tuples whose capital loss attributes are 0 or NULL. That leaves us with 1427 tuples. The range of capital loss in these tuples is from 155 to 3900, with 89 distinct values.

We also conduct experiments on a synthetic data set, which uses the same schema as the Adult Database, to adjust a variety of parameters and comprehensively evaluate the permutation-based approach. We populate the database with different numbers of tuples, distributions of the capital loss attribute, and correlations between the capital loss attribute and quasi-identifiers. Details of the synthetic data set will be described with the experimental results.

The main experiments we report on in this paper focus on *query answering accuracy*. In this set of experiments, we compare the accuracy of the bounds derived from the generalized table and the permuted table. Specifically, let l and u be a lower and an upper bound of an aggregate query result r respectively. We define $error = (u - l)/r$ to be the relative error of the bounds. The smaller $error$ is, the more accurate the bounds are. We also compare the accuracy of those bounds when using different optimization criteria to get (k, e)-anonymous partitions.

We also designed experiments to study *query answering overhead*. As described previously, to answer an aggregate query Q over a permuted database, we first rewrite it into a query Q' which queries the permuted and the help tables. These experiments measure the running time of the rewritten query, and demonstrate that it is quite reasonable compared with the time that it takes to execute Q over the un-permuted microdata table. Details of these experiments can be found in the full version of this paper [5].

7.2 Accuracy: Generalization vs Permutation

We first compare the relative errors of the bounds derived from the generalized table and the permuted table, given the same partition. Specifically, the partition, which satisfies $(4, 0)$-anonymity (the same as 4-diversity in the ℓ-diversity work), is obtained by using the ℓ-diversity algorithm and the same generalization hierarchies reported in [8].[1] The resulting 4-diverse partition is composed of 25 groups.

[1] We note that the experiments in the work of ℓ-diversity actually computed 6-diverse partitions. However, after generalization using those 6-diverse partitions, a majority of quasi-identifiers (6 out of 8) including "age", "race", "native country", etc., are all generalized to "*", which essentially removes these attributes from the microdata table. To make the

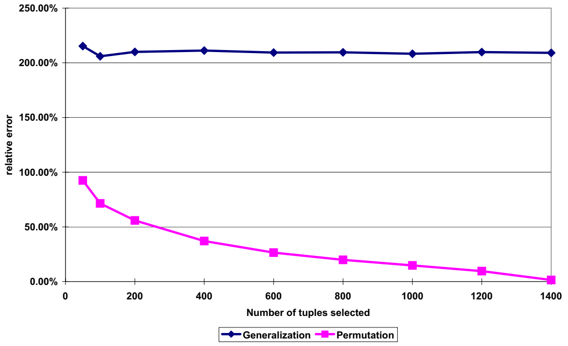

Figure 5. The relative errors of arbitrary queries when using the generalized table and the permuted table respectively

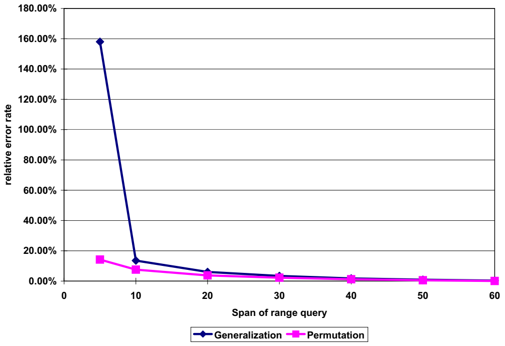

Figure 6. The relative errors of range queries when using the generalized table and the permuted table

We first consider a general model of aggregate queries. Note that no matter what the query condition is, the result of the condition is to select a set of tuples in each group of the partition. Therefore, a query can be viewed as $\{T_1, \ldots, T_r\}$, where each T_i is a subset of $\mathcal{D}_i$. By selecting different tuples in each group, we may model arbitrary aggregate queries.

In the first experiment, we issue queries that randomly touch an arbitrary number of tuples from the table, and compute the average of their sensitive attribute values. We call these queries *arbitrary queries*. Figure 5 shows that the relative errors introduced by the generalized table is significantly higher (often over twice the actual query results) than that by the permuted table. Further, as the total number of tuples selected increases, the relative error introduced by the permuted table drops dramatically, while that introduced by the generalized table does not drop at all.

Since range queries are common in practice, in the next experiment, we issue aggregated queries over a certain range of the age attribute. In particular, we issue a sequence of queries of the form “select avg(capital_loss) from adult-table where age $\geq$ X and age $\leq$ Y”, and vary the range $[X, Y]$. The relative errors for range queries are shown in figure 6. We see that the accuracy of the bounds derived from both the generalized table and the permuted table improve for range queries. For the generalized table, the error is still very high when the span of the range query is small, because some groups are only partially covered by the query. When the span of the range query increases, more and more groups are completely covered by the query. Thus, we observe a dramatic drop of relative errors for the generalized table. However, the error for the generalized table is still always higher than that for the permuted table.

The adult database contains 9 attributes as quasi identifiers. When we have fewer attributes, the generalization hierarchy may be less coarse and lead to smaller partitions. To study the impact of dimensions on accurate query answering, we have also conducted experiments when assuming “age” is the only quasi identifier. We have observed very similar trends to that in Figures 5 and 6. Due to space limitations, we do not show the results of this experiment. Our experiments suggest that poor query answering accuracy from the generalized table is intrinsic to the domain generalization approach, and not caused by the high dimensionality of quasi-identifiers.

7.3 Accuracy: Partitioning Strategies

The way the microdata table is partitioned has a great impact on the accuracy of the bounds derived from the permuted table. Next, we compare the relative errors of permuted tables when using partitions generated by the following algorithms (see section 6): (1) ℓ-diversity based on domain generalization; (2) optimization algorithm 1: the approximation algorithm for the minimum max-of-error problem; (3) optimization algorithm 2: the optimal algorithm for the minimum sum-of-error problem; and (4) a random algorithm: this algorithm sequentially scans each tuple and as long as the scanned tuples have k distinct sensitive attribute values, and its range is no less than e, they form a group of the partition. The random algorithm serves as a baseline for comparison. We set $k = 4$ as before, and set $e = 100$ for the two optimization algorithms and the random partitions. To favor the generalization-based approach, the groups in the partition generated by ℓ-diversity algorithm is only required to have no less than 4 distinct values. The parameters of the

comparison more meaningful and in favor of the generalization-based approach, we instead choose 4-diverse partitions so that interesting attributes such as “age” can be generalized to reasonable domains. In fact, to prevent “age” from being generalized to “*” when using the ℓ-diversity algorithm, we also have to remove some outliers (6 tuples with age over 80). Otherwise, the “age” attribute will still be suppressed even with 4-diversity.

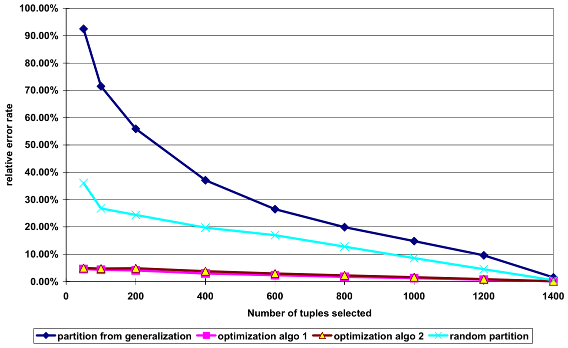

Figure 7. The relative errors of arbitrary queries when using different partitioning algorithms

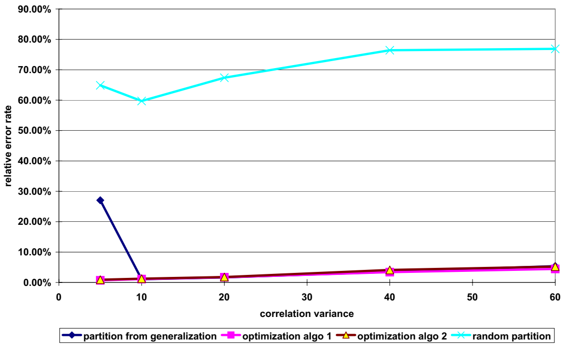

Figure 8. The relative errors of range queries using partitions from different partitioning algorithms, when the strength of the correlation between quasi-identifiers and the sensitive attribute varies.

arbitrary queries and the range queries in this experiment are the same as in the first two experiments.

The relative errors corresponding to the above algorithms are shown in figure 7 for arbitrary queries (the figure for range queries is omitted due to space). It is clear that the two optimization algorithms introduce significantly less relative errors than the other two algorithms. This can be easily explained since the optimization algorithms are not constrained by pre-defined domain hierarchies. They have more flexibility to partition the table and achieve better accuracy. We also observe that the partition derived from predefined generalization hierarchies is even worse than the random partition for arbitrary queries.

7.4 Accuracy: Correlations

Intuitively, if there is a strong correlation between quasi-identifiers and the sensitive attribute, the tuples in the same group tend to have similar values in the partition generated through domain generalization. This may result in more accurate bounds for answering range queries. Our next experiment is to investigate the impacts of correlation on the accuracy of query answering. We compare the partitions obtained by the above four algorithms when varying the correlation between quasi-identifiers and the sensitive attribute. We run range queries that select tuples whose age attributes are in the range $[X, Y]$ where $Y - X = 30$. In the synthetic data set, we introduce a correlation between "age" and "capital loss". The larger a tuple's age attribute, the larger its capital-loss, with a certain variance, which controls the strength of the correlation.

Figure 8 shows the relative errors of the four algorithms when the strength of correlation varies. We observe that as the correlation is strong(variance=5), tuples with the same age often have the same capital loss. Thus a higher generalization is needed, which causes the partition from generalization to yield a large error. As the variance goes up to 10, a lower generalization is sufficient since it is more likely for tuples in the same group to have different capital loss values. That explains the quick drop of the error when variance=10. As the variance keeps increasing, tuples in the same group tends to have quite different sensitive attribute values, which will cause the error to increase. Since "age" is the only quasi identifier in the synthetic data set, a range query may completely cover many groups in the partition obtained through domain generalization. Therefore, it yields comparable accuracy with the partitions generated by the two optimization algorithms. The randomly algorithm does not take advantage of the correlation between "age" and "capital" loss, and thus performs poorly as expected.

Finally, we have also studied the tradeoff between privacy and query answering accuracy. Intuitively, the larger k and e are, the more tuples each group in a (k, e)-anonymous partition tends to include, which will in turn introduce more errors when answering aggregate queries. We have conducted experiments to see the change of query answering accuracy when varying k and e respectively. We observe that when fixing $e = 50$ and vary k from 4 to 40, the relative errors of the two optimization algorithms do not deteriorate much as we increase k. This suggests that the optimization algorithms are capable to generate partitions that preserve high privacy while still supporting accurate aggregate query answering. Similar observations can be found when varying e. We omit these figures due to space limits.

8 Related Work

The privacy vulnerability of the release of de-identified microdata was first discussed by Sweeney [11], who also proposed k-anonymity as a model for protecting privacy of microdata. The concept of ℓ-diversity is introduced by Machanavajjhala et al. [8] to guard against attackers with background knowledge. There has been considerable recent work on k-anonymity, including finding optimal generalizations, approximation algorithms, pruning techniques, etc., to reduce the imprecision in microdata (see e.g., [2, 4, 6]. In these proceedings, Li et al. [7] also address the anonymization problem for the case of numerical attributes. They propose the notion of t-closeness, which requires that the distribution of a sensitive attribute in a group is close to the distribution of that attribute in the overall table. Their framework is also based on generalization of quasi-identifiers. However, none of these previous works discuss their impact on the accuracy of answering aggregation queries.

Permutation-based anonymization is a natural generalization of data swapping techniques where privacy is achieved by exchanging the sensitive attributes of pairs of randomly selected records [1]. Since data swapping is done globally, it has a much larger impact on microdata utility. Even when done in a controlled manner (e.g., rank-based data swapping [9]), it still produces large errors for aggregate queries, as shown in [5]. Data swapping also has limitations in terms of privacy protection. For example, assuming an attacker knows a particular tuple's sensitive attribute (e.g., his own record) and its value is unique in the table, then he may be able to derive the exact attribute of another tuple. In this sense, data swapping offers privacy similar to that of 2-anonymity. The permutation-based approach, which satisfies (k, e)-anonymity, forces the attacker to acquire at least $k - 1$ records' sensitive attributes in order to reveal one record's exact value.

Perturbation-based approaches [1, 3] have also been proposed for privacy, where noise following a certain distribution is added to the sensitive attribute of each tuple. Such an approach inevitably changes important statistics of the marginal distributions of sensitive attributes (e.g., variance) [10]. Further, depending on the distribution of added noise (e.g., Gaussian Distribution), it is often difficult to derive deterministic bounds for answering aggregate queries.

Recently, Xiao et al. [13] propose to achieve k-anonymity by separating quasi-identifiers and sensitive attributes into two tables, connected by the group ID of each tuple. It is easy to see that their scheme is equivalent to a permutation of sensitive attributes among tuples in the same group. However, as in most other works on k-anonymity, this work focuses only on categorical sensitive attributes, and their techniques cannot be directly applied to handle numerical sensitive attributes.

9 Conclusion

We proposed a privacy goal to better capture the protection of numeric sensitive attributes in microdata. We also presented permutation based anonymization, and showed that we can achieve the same privacy guarantees as generalization-based techniques, but answer aggregation queries much more accurately.

There are many interesting issues to be explored in the future. In particular, we would like to investigate the use of permutations under other privacy objectives, identify efficient optimal or approximation algorithms under such privacy objectives, and study their impact on the accuracy of answering aggregation queries.

References

[1] N. Adam and J. Wortman. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21(4), 1989.

[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. P. raby, D. Thomas, and A. Zhu. k-Anonymity: Algorithms and Hardness. Technical report, Stanford University, 2004.

[3] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In *Proceedings of the ACM SIGMOD International Conference on M anagement of Data*, Dallas, Texas, May 2000.

[4] R. J. Bayardo and R. Agrawal. Data Privacy through Optimal k-Anonymization. In *Proceedings of 21st IEEE International Conference on Data Engineering*, Tokyo, Japan, Apr. 2005.

[5] N. Koudas, D. Srivastava, T. Yu, and Q. Zhang. Aggregate Query Answering on Anonymized Tables. Technical Report TR-2006-16, North Carolina State University, 2006.

[6] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito - Efficient Full-Domain K-Anonymity. In *In Proceedings of ACM SIGMOD Conference*, Baltimore, MD, June 2005.

[7] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and ℓ-Diversity. In *Proceedings of IEEE International Conference on Data Engineering*, 2007.

[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-Diversity: Privacy Beyond k-Anonymity. In *IEEE International Conference on Data Engineering*, Atlanta, GA, Apr. 2006.

[9] R. Moore. Controlled Data-Swapping Techniques for Masking Public Use Microdata Sets. Technical Report Statistical Research Division Report Series, RR 96-04, US Bureau of Census, 1996.

[10] K. Muralidhar and R. Sarathy. Security of Random Data Perturbation Methods. *ACM Transactions on Database Systems*, 24(2), 1999.

[11] L. Sweeney. Guaranteeing Anonymity When Sharing Medical Data, the Datafly System. *Journal of the American Medical Informatics Association*, pages 51–55, 1997.

[12] U.C.Irvin Machine Learning Repository. http://www.ics.uci.edu/m̃learn/mlrepository.html.

[13] X. Xiao and Y. Tao. Anatomy: Simple and Effective Privacy Preservation. In *International Conference on Very Large Data Bases*, Seoul, Korea, Sept. 2006.

[14] C. Yao, S. Wang, and S. Jajodia. Checking for k-Anonymity Violiation by Views. In *International Conference on Very Large Data Bases*, Trondheim, Norway, Aug. 2005.