Distribution-based Microdata Anonymization

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

Divesh Srivastava
AT&T Labs – Research
divesh@research.att.com

Ting Yu
North Carolina State University
yu@csc.ncsu.edu

Qing Zhang
Teradata
qzhangqing@gmail.com

ABSTRACT

Before sharing to support ad hoc aggregate analyses, microdata often need to be anonymized to protect the privacy of individuals. A variety of privacy models have been proposed for microdata anonymization. Many of these models (e.g., t-closeness) essentially require that, after anonymization, groups of sensitive attribute values follow specified distributions. To support such models, in this paper we study the problem of transforming a group of sensitive attribute values to follow a certain target distribution with minimal data distortion. Specifically, we develop and evaluate a novel methodology that combines the use of sensitive attribute permutation and generalization with the addition of fake sensitive attribute values to achieve this transformation. We identify metrics related to accuracy of aggregate query answers over the transformed data, and develop efficient anonymization algorithms to optimize these accuracy metrics. Using a variety of data sets, we experimentally demonstrate the effectiveness of our techniques.

1. INTRODUCTION

Cooperation among enterprises and governmental agencies often requires the sharing of private microdata, because they support flexible and ad hoc aggregate analyses, a major advantage over precomputed aggregates. Due to privacy concerns, microdata need to be properly transformed to prevent the discovery of any individual's sensitive attributes. A variety of privacy models and techniques have been proposed to address this problem. k-anonymity [9, 10], the first privacy model proposed for microdata anonymization, only requires that for each tuple t there exist at least $k-1$ other tuples with the same quasi-identifier as t. Since k-anonymity does not consider the distribution of sensitive attributes, it suffers from the homogeneity attack when all these k tuples have the same or similar sensitive attribute values [7]. Several privacy models have been recently proposed to remedy this problem, including ℓ-diversity [7] and t-closeness [6].

Though these privacy models have different definitions or target different types of sensitive attributes, they essentially specify constraints on the sensitive attribute distribution of a group of individual tuples. For example, ℓ-diversity requires that there are at least ℓ well-represented sensitive attribute values in P. Similarly, t-closeness requires that the distribution P of sensitive attributes in a group should be similar to that of the whole microdata table.

Given these distribution-based privacy models, the anonymization problem is thus to transform the original microdata table to groups of tuples so that the sensitive attribute distribution of each group satisfies the privacy goal. Existing approaches are to form groups (through quasi-identifier generalization or clustering) such that the distributions of those groups naturally satisfy the privacy goal without any modification. However, there are two concerns with such approaches. First, there may not always exist groupings that satisfy the privacy goal. For example, even if the microdata table as a single group satisfies ℓ-diversity, no non-trivial grouping may do so. Second, the data owner has no control over how the final grouping will look. Though these groups after anonymization satisfy a privacy goal, they may not be suitable for the target research analysis. For example, for health care research, it may be desirable to group patient records by age group, by geography, etc. However, to achieve a privacy goal, we may have to put an arbitrary subset of patient records into a group. We observe that, with existing approaches, there is a dilemma between the desirable groupings for microdata analysis and those for privacy protection.

The essential reason for the above dilemma is that existing approaches solely rely on the original distribution of a group to achieve a privacy goal. In other words, they tightly couple grouping with privacy goals. In this paper, we propose *distribution transformation* as a new methodology to support distribution-based anonymization. Specifically, we study the following problem: given an arbitrary group of tuples with quasi-identifiers and a sensitive attribute and a target distribution on the sensitive attribute that satisfies a privacy model, modify the sensitive attribute values of tuples in the group such that one can only infer from the anonymized data that for each tuple its sensitive attribute in the group follows the target distribution but nothing more. We call this problem the *distribution transformation problem*.

One major advantage of this problem setting is that it clearly decouples grouping from privacy goals. It allows data owners to come up with more useful groupings for data analysis while still protecting privacy. For example, both ℓ-diversity and t-closeness require k-anonymity. Using our problem setting, we may first use any basic k-anonymity algorithm to come up with groupings that minimize the generalization. Then for each group, we can further achieve ℓ-diversity or t-closeness (i.e., define target distributions that satisfy ℓ-diversity or t-closeness). This problem setting further

VLDB '09, August 24-28, 2009, Lyon, France

offers an additional flexibility to have different target distributions for different groups, so that the data owner can strike a fine-grained tradeoff between privacy and data utility. It is also easy to see that the distribution transformation problem is a more general problem setting, in the sense that it is independent of specific privacy goals. Solutions to this problem thus can be generally applied to existing distribution-based privacy models.

As is true for any anonymization technique, besides privacy protection, solutions to the distribution transformation problem should also maintain as much useful information as possible in the published microdata. In particular, it should still be possible to answer ad hoc aggregate queries with reasonable accuracy. Otherwise, the whole purpose of data sharing is lost. For instance, one naive approach is to replace each record's sensitive attribute value with one that is randomly generated following the target distribution. However, this approach changes the original data completely, and will have a significant negative impact on the accuracy of ad hoc query answering. In particular, it is impossible to get deterministic bounds except the trivial ones for aggregate queries. This is because by only looking at the perturbed microdata table each tuple's true sensitive attribute can take any value *independently* in the domain. To get deterministic bounds, we have to consider the worst case which will result in trivial bounds. In this paper, we propose a methodology for distribution-based microdata anonymization that returns accurate bounds as answers to ad hoc aggregate queries, and make the following contributions.

First, we present a general framework for distribution-based microdata anonymization. We develop novel techniques that combine sensitive attribute generalization and permutation to transform microdata such that each record appears to follow a target distribution.

Second, we identify metrics related to accuracy of query answers over the anonymized microdata, and develop algorithms to optimize these metrics. Intuitively, the less generalization we perform on sensitive attributes, the more accurate query results we tend to obtain. We design efficient algorithms to minimize the sum of ranges of sensitive attribute generalizations.

Third, to further support accurate aggregate query answering, we propose to add fake sensitive attribute values into the original data set before performing sensitive attribute generalization and permutation. The use of fake values adds significant challenges to the anonymization process. We design an efficient algorithm to compute the optimal fake values that minimize our proposed metrics.

Finally, we conduct comprehensive experiments over both real and synthetic data sets to verify the advantages of the proposed techniques. The experimental results show that microdata protected through sensitive attribute generalization and permutation combined with the use of fake sensitive attribute values can answer ad hoc queries with reasonable accuracy. Our experiments also verify the effectiveness of the proposed optimization algorithms.

We organize the rest of the paper as follows. In Section 2, we use examples to show the basic idea of our approach. In Section 3, we present a general privacy protection framework for distribution-based microdata anonymization. An optimal algorithm for sensitive attribute generalization is presented in Section 4. We discuss the use of fake values in Section 5. Experimental results are reported in Section 6. We discuss closely related work in Section 7 and conclude this paper in Section 8.

2. ILLUSTRATIVE EXAMPLES

Figure 1 shows an example employee record table. Before this table is shared, the explicit identifier attribute Name should be removed. Attributes Zipcode and Gender are quasi-identifiers and Salary is the sensitive attribute. For simplicity, we assume the domain of Salary to be $\{30K, 40K, 50K, 60K\}$.

	ID	Quasi-identifiers		Sensitive
tuple ID	name	zipcode	gender	salary
1	Alice	91110	F	$30K
2	Bob	91110	M	$40K
3	Carol	91110	M	$50K
4	Debra	91130	F	$60K
5	Elaine	91210	F	$40K
6	Grace	91220	F	$30K
7	Helen	91240	F	$50K
8	Jason	91310	M	$40K
9	Kyle	91320	M	$60K
10	Leo	91330	M	$60K
11	Nancy	91340	F	$60K

Figure 1: An example microdata table

		Quasi-identifiers		Sensitive
group ID	tuple ID	zipcode	gender	salary
1	1	91110	F	$40K
1	2	91110	M	$60K
1	3	91110	M	$30K
1	4	91130	F	$50K
2	5	91210	F	{$30K,$40K}
2	6	91220	F	{$50K, $60K}
2	7	91240	F	{$30K, $40K, $50K, $60K}
3	8	91310	M	{$30K, $40K, $50K, $60K}
3	9	91320	M	{$30K, $40K, $50K, $60K}
3	10	91330	M	{$50K, $60k}
3	11	91340	F	{$30K, $40K}

Figure 2: An example microdata table after sensitive attribute generalization and permutation

Suppose the data owner wants to share this information with a third party for business analysis, which prefers to have tuples grouped according to the first three digits of their Zipcode. For privacy protection, it is further required that the salary attribute values in each group should satisfy 4-diversity. To achieve this privacy goal, the data owner may specify for each group a distribution (called a target distribution) over the domain of Salary that satisfies 4-diversity. For ease of explanation, we assume the target distribution for each group is the uniform distribution. In practice, each group may have its own target distribution.

Clearly, except the first group (those with zipcode 911**), the distributions of the sensitive attribute values of the other two groups do not follow the target distribution. In this paper, we study the following problem: given a target distribution P of sensitive attribute values for a group of tuples, perform anonymization so that for each individual tuple t one can only infer from the anonymized data that t's sensitive attribute value follows P but nothing more. We call it the *distribution transformation* problem.

This problem cannot be solved by existing anonymization techniques (e.g., quasi-identifier generalization or permutation), as none of them allows the modification of sensitive attribute values, and thus cannot change the original distribution of a group to fit into the target distribution.

Our basic scheme is to leave quasi-identifiers unchanged and generalize and permute the sensitive attribute values (see section 3). Figure 2 shows one possible anonymized microdata table after applying our scheme. Figure 3 shows the generalization hierarchy for the Salary attribute. As we show in Section 3.1, any target distribution can be specified over the domain generalization hierarchy by giving different weights to the children of nodes in the hierarchy tree. The weight assignments in Figure 3 correspond to a uniform distribution. Note that in group 1, since its distribution is already the same as the target distribution, no generalization is needed. Instead, to protect privacy, we only need to perform a random per-

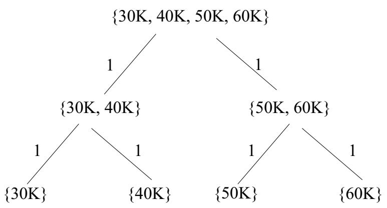


Figure 3: An example domain generalization hierarchy

mutation to break the association between a tuple and a particular sensitive attribute value. For the other two groups, both generalization and permutation are needed to achieve the privacy goal.

Once a group of tuples and the corresponding target distribution are given, each tuple's sensitive attribute is generalized to a subdomain corresponding to a node in the domain hierarchy. Then the generalized domains of the tuples in the same group are permuted. For example, in group 1 of Figure 2, each tuple's sensitive attribute is generalized to itself. These values are then permuted among group 1. In group 2, the three sensitive attribute values 30K, 40K and 50K are generalized to {30K, 40K}, {30K, 40K, 50K, 60K} and {50K, 60K} respectively. Then again we perform a random permutation of the generalized sensitive attributes among group 2. The combination of generalization and permutation guarantees that even if an attacker can associate a tuple with a unique individual through its quasi-identifiers, he can only infer that a tuple's sensitive attribute follows the target distribution.

Since sensitive attributes are generalized and permuted, we usually cannot get exact answers to aggregate queries. But we can always derive a correct bound for a query from the anonymized table. For example, consider the following query: What is the average salary of female employees?

As our scheme does not change the quasi-identifiers of each tuple, we know exactly how many tuples in each group are selected by the query, i.e., 2 tuples from group 1, 3 tuples from group 2, and 1 tuple from group 3. Since sensitive attributes are generalized and permuted, we do not know these tuples' original salaries. But we can derive the lower and upper bounds of the query result. Specifically, from group 1, the minimum and the maximum of the sum of the two tuples' salaries are 30K+40K = 70K and 50K+60K=110K, respectively. For group 2, consider the generalized value {30K, 40K}. Since we do not know its original value, we have to estimate conservatively to make sure that the derived bounds are correct. Thus, the minimum and the maximum of the sum of the three tuples in group 2 would be 30K+30K+50K=110K and 40K+60K+60K=160K respectively. And the bounds for the single tuple from group 3 would be [30K, 60K]. Overall, the bounds for the answer of the query would be [35K, 55K].

Domain generalization and permutation ensure that each individual tuple's sensitive attribute value follows the target distribution. Meanwhile, compared with perturbation-based approach, the technique preserves some of the properties of sensitive attributes in the whole group. For example, from group 1, one can only know that for each tuple its salary has equal chance to be any of the four possible values, i.e., it follows the target distribution. But one also knows that inside this group, there is exactly one occurrence of each value in the domain. This helps one obtain more accurate deterministic bounds for aggregate queries.

Note that if sensitive attribute values were randomly generated to follow the target distribution, we could get inaccurate answers to queries without any confidence guarantee. For example, if the four tuples with group ID = 3 were associated with the randomly generated salary values 30K, 40K, 50K and 60K (following the target uniform distribution), the query "What is the average salary of people in the 913* zipcode" would return the incorrect value 45K, whereas our method based on generalization and permutation would return the range [35K, 55K], which includes the correct value of 55K. If one wanted to provide deterministic bounds based on the randomly generated salary values, one could only provide the trivial range [30K, 60K].

		Quasi-identifiers		Sensitive
group ID	tuple ID	zipcode	gender	salary
1	1	91110	F	{$30K, $40K}
1	2	91110	M	$50K
1	3	91110	M	{$30K, $40K}
1	4	91130	F	$60K
2	5	91210	F	{$50K, $60K}
2	6	91220	F	{$30K, $40K}
2	7	91240	F	{$30K, $40K, $50K, $60K}
3	9	91310	M	{$30K, $40K, $50K, $60K}
3	10	91320	M	{$30K, $40K, $50K, $60K}
3	11	91330	M	{$30K, $40K, $50K, $60K}
3	12	91340	F	{$30K, $40K, $50K, $60K}

Figure 4: An example microdata table after sensitive attribute generalization and permutation, where aggregate queries cannot be accurately answered.

Given a group of sensitive attribute values, there are many ways to generalize them to reach the target distribution. For example, Figure 4 shows another way of generalization and permutation. An extreme example is to generalize each sensitive attribute to the whole domain. Though they offer the same privacy, the accuracy of query answering from these different schemes may vary greatly. We can verify that the bounds derived from the trivial generalization table are much worse than those from the table in Figure 2.

Intuitively, the less generalization we perform, the better bounds we tend to obtain. Based on this observation, in Section 4, we identify optimization metrics and design algorithms that produce sensitive attribute generalization and permutation schemes that offer better query answering accuracy.

A second important technique we propose to improve the accuracy of query answering is to introduce fake sensitive attribute values along with sensitive attribute generalization and permutation. To show the benefit of this technique, let us have a closer look at group 2 in Figure 2. With only generalization and permutation, the generalization shown in Figure 2 is the best we can do to minimize the sum of ranges of generalizations. If we ask what is the average salary of employees in area with zipcode 912**, the best bound we can get is $[110K/3, 160K/3]$.

Instead, if we introduce another sensitive attribute value $60K$ into the group (we call it a *fake value* as it does not appear in the original data set), the resulting distribution will be exactly the same as the target distribution. Thus we only need to perform a permutation without any generalization. Group 2 will then be as follows.

		Quasi-identifiers		Sensitive
group ID	tuple ID	zipcode	gender	salary
2	5	91210	F	{$30K}
2	6	91220	F	{$40K}
2	7	91220	F	{$50K}
				{$60K}

Here we list three employees in this group but four possible salaries, meaning that their salaries can be any three out of the four.[1] From the above table one can still only infer that the possible

[1] This table can be easily implemented by separately creating a quasi-identifier table and a sensitive attribute table, which can be joined through the group ID, as in [13].

salary of each employee is uniformly distributed between $30K$ and $60K$. Meanwhile, since we reduce the generalization of sensitive attributes dramatically, the average salary of employees in that area can be more accurately bounded to be $[40K, 50K]$.

From the above example, we see that adding fake values may reduce the level of generalizations needed to match a target distribution. Meanwhile, fake values also introduce uncertainty into the original data set. Without any restriction, given any data set, we can always protect privacy without using any generalization. But this may not be the best choice for accurate aggregate query answering. For example, in group 3 of Figure 2, we can add the following fake values: three $30K$, two $40K$ and three $50K$. Clearly, this will result in much worse bounds for aggregate queries. For example, the only bounds we can get for the average salary for group 3 would be $[130K/4, 230K/4]$, which is worse than $[140K/4, 220K/4]$, the bound when only applying generalization and permutation. Thus, it would be sensible to minimize the sum of ranges of generalizations while limiting the number of added fake values. In Section 5, we present an efficient algorithm to do so.

3. A FRAMEWORK FOR DISTRIBUTION-BASED ANONYMIZATION

3.1 Basic Concepts and Notations

As in most works on microdata anonymization, we assume the schema of a de-identified microdata table consists of two parts: a set of quasi-identifiers $\{Q_1, \ldots, Q_k\}$, and a sensitive attribute S. Our technique can be used to protect both numerical and categorical sensitive attributes. Here we focus on numerical data as they support more interesting aggregation operations, such as SUM and AVERAGE, while for categorical data usually only COUNT is meaningful. For simplicity, we also assume the domain of S is finite.

Given the domain $\mathcal{D}$ of S, we say $\mathcal{D}' \subset \mathcal{D}$ is a subdomain of S if it covers a continuous segment of $\mathcal{D}$. In other words, there exists no $t \in \mathcal{D}(S)$ such that $t \notin \mathcal{D}'$ but $min(\mathcal{D}') \leq t \leq max(\mathcal{D}')$. We thus also denote $\mathcal{D}'$ as $[min(\mathcal{D}'), max(\mathcal{D}')]$, and call $max(\mathcal{D}') - min(\mathcal{D}')$ the range of $\mathcal{D}'$.

A generalization hierarchy of S is a rooted tree H that satisfies the following conditions: (1) Each node of H is a subdomain of $\mathcal{D}$; (2) $\mathcal{D}$ is the root; (3) All the children of a node N form a partition of N; and (4) A leaf node is a subdomain $\{t\}$, where $t \in \mathcal{D}$.

As mentioned in Section 2, the owner of a microdata table specifies a partition $\{g_1, \ldots, g_n\}$ of the microdata table and a target distribution P_i for each group g_i. The requirement of anonymization is to make sure that in the released microdata table one can only infer that the sensitive attribute values of each group g_i follow P_i. Note that this is a very flexible model for microdata protection. In particular, it does not require the target distribution to be the same for all groups. For example, the target distributions P_i and P_j may both satisfy ℓ-diversity or t-closeness, but are different such that each is closer to the original distributions of g_i and g_j. By doing so, we may reduce data distortion for privacy protection.

Without loss of generality and for ease of explanation, in the rest of our discussion, we assume that the data owner only specifies a target distribution for the whole microdata table. When we have multiple groups each with their own target distributions, we can simply use the techniques in this paper to accommodate the target distribution of each group separately.

From an anonymized microdata table T, if we can only derive that the distribution of each tuple's sensitive attribute follows a distribution P, we say T preserves privacy in terms of P, or is P-private for short. For example, the table in Figure 2 is U-private, where U is the uniform distribution over {30K, 40K, 50K, 60K}.

Note that there are two types of privacy in microdata anonymization. One is existence privacy, where the existence of a record for an individual is considered sensitive. The other is linkage privacy, where the association between a tuple and its sensitive attribute values is sensitive. Depending on specific applications, either one of them or both may be desirable. Like many existing works, in this paper we focus on linkage privacy [8, 13, 15].

Given a target distribution P and a domain generalization hierarchy H for S, we assign a weight to each edge of H such that for any two children C_1 and C_2 of a node N, we have $weight((N, C_1)) : weight((N, C_2)) = P(x \in C_1) : P(x \in C_2)$. We call the resulting domain hierarchy a *privacy-annotated* hierarchy. For simplicity, we assume all the weights are integers. It is easy to see that a privacy-annotated hierarchy represents a distribution over $\mathcal{D}$. For example, Figure 3 is a privacy-annotated hierarchy that corresponds to a uniform distribution. In the rest of this paper, unless otherwise noted, we assume that a target distribution is given as a privacy-annotated hierarchy.

3.2 Sensitive Attribute Generalization and Permutation

As illustrated earlier, the basic idea of our approach is to generalize the sensitive attribute value of each tuple to a subdomain of $\mathcal{D}$. We then randomly permute the resulting subdomains among all the tuples, so that the distribution of the possible value of a tuple t's sensitive attribute $t.S$ is the same as the target distribution. Note that the distribution of $t.S$ depends on the given privacy-annotated hierarchy. For example, for a subdomain {30K, 40K}, given the privacy-annotated hierarchy in Figure 3, we know that, if $t.S \in$ {30K, 40K}, then $t.S$ has an equal probability to be either 30K or 40K. On the other hand, suppose the ratio between the weights of 30K and 40K is 2:1 instead of 1:1 in the hierarchy (i.e., the hierarchy corresponds to a different target distribution). If $t.S \in$ {30K, 40K}, the probabilities of $t.S$ to be 30K and 40K will be 2/3 and 1/3 respectively. Formally, we have the following definitions.

DEFINITION 3.1. *Let g be a multiset of elements in $\mathcal{D}$ and G be a multiset of subdomains of $\mathcal{D}$.*[2] *We say G is a* generalization *of g if there is a one-to-one mapping f from g to G such that $\forall s \in g, s \in f(s)$.*

DEFINITION 3.2. *Let $G = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a generalization of g, and P be a target distribution over $\mathcal{D}$. We say G is P-*private generalization *of g if for any element $t \in \mathcal{D}$, $P(x = t) = \frac{1}{n}\sum_{1 \leq i \leq n} P(x = t | x \in \mathcal{D}_i)$.*

Intuitively, if G is a P-private generalization of g, then from a random permutation of G the probability for any element in $\mathcal{D}$ to be in g follows P. Thus, G effectively hides the original distribution of values in g. From G, an attacker can only derive that elements in g follow P but nothing more.

For example, let U denote the uniform distribution over {30K, 40K, 50K, 60K}. Then both {{30K}, {40K}, {50K-60K}, {50K-60K}} and {{30K, 40K}, {30K, 40K}, {50K}, {60K}} are U-private generalizations of {30K, 40K, 50K, 60K}, and the latter is also a U-private generalization of {30K, 30K, 50K, 60K}.

Since the interpretation of a generalized subdomain is subject to the target distribution, given in the form of a privacy-annotated hierarchy with weights on each subdomain, a trivial P-private generalization is to generalize every sensitive attribute value to the whole

[2] g is a multiset because multiple tuples may have the same sensitive attribute values. Similarly, G is a multiset since multiple elements in g may be generalized to the same subdomain.

domain $\mathcal{D}$. For example, suppose the target distribution is a normal distribution. Then if a sensitive value is generalized to the root domain, one can only infer that it follows the target normal distribution (not a uniform distribution). Though secure, the resulting microdata table from this trivial generalization is hardly useful. Thus, it is important to develop algorithms to produce generalizations which protect privacy, and further, can answer aggregate queries accurately. Before presenting such algorithms, we first briefly discuss how to answer aggregate queries over an anonymized microdata table.

3.3 Answering Aggregate Queries

In this paper, we consider ad hoc data analysis in OLAP and focus on queries that select a subset of tuples based on *arbitrary conditions* on quasi-identifiers and then aggregate over the sensitive attribute, i.e., the aggregate queries are of the form "SELECT Aggr(S) FROM Microdata WHERE C", where C is a condition on quasi-identifiers, and Aggr may be common aggregates such as SUM, AVERAGE and MIN.

Given an aggregate query, since quasi-identifiers are not changed during anonymization, from the anonymized table, we know exactly how many tuples in each group are selected by the query. Thus, from a P-private generalization of the group, we can estimate the lower and upper bounds of the aggregation inside each group. These bounds can then be combined together to form those for the whole table. In fact, to get the bounds for an aggregate query, what matters is only the number of selected tuples in each group. Therefore, as in [15], we can pre-compute for each group the bounds of each type of aggregates when different numbers of tuples in that group are selected by a query, and store these bounds in an auxiliary table. When answering an aggregate query Q, after knowing the number of tuples selected by Q, we only look up the auxiliary table, and combine together the bounds of each group to get the bounds for Q. This process can be easily implemented using SQL. Therefore, there is no need to modify the DBMS's query engine to support aggregate queries over the anonymized table.

4. OPTIMAL ALGORITHM FOR SENSITIVE ATTRIBUTE GENERALIZATION

As mentioned before, a trivial P-private generalization that generalizes every sensitive attribute value to the whole domain is not useful for answering aggregate queries. In this section, we present optimization criteria for distribution-based anonymization so that aggregate queries can be more accurately answered.

4.1 Minimize the Sum of Ranges of Sensitive Attribute Generalizations

The higher a subdomain is in the domain hierarchy, the more general it is, and thus the more "uncertainty" is introduced when answering queries. A natural measure of the uncertainty of a subdomain is its range, i.e., the difference between the maximum and the minimum elements in the subdomain. Formally, we define the following optimization problem.

PROBLEM 4.1. *Given a multiset g of sensitive attribute values, a privacy-annotated domain generalization hierarchy H that corresponds to a distribution P over $\mathcal{D}$, find a P-private generalization G of g such that $\sum_{\mathcal{D}' \in G} range(\mathcal{D}')$ is minimized.*

We call G the optimal P-private generalization of g.[3]

To facilitate our discussion, we introduce the following notations. Given a subdomain $\mathcal{D}'$ and a multiset g of sensitive attribute values, we denote $g_{\mathcal{D}'}$ to be the multiset that contains all the elements in g that belong to $\mathcal{D}'$. For example, suppose $g = \{30K, 30K, 40K, 50K\}$ and $\mathcal{D}' = \{30K, 40K\}$. Then $g_{\mathcal{D}'} = \{30K, 30K, 40K\}$.

Further, let $N = (n_1, \ldots, n_k)$ be a sequence of integers, $W = (w_1, \ldots, w_k)$ be a sequence of weights, and m be an integer. We say $(n'_1, \ldots, n'_k)$, where n'_i is an integer and $n'_i \leq n_i$ for $i = 1, \ldots, k$, is an m-bound allocation of N subject to W, if $n'_1 : n'_2 : \cdots : n'_k = w_1 : w_2 : \cdots : w_k$, and $\sum_{1 \leq i \leq k} n'_i \leq m$ (i.e., proportional). We say $(n'_1, \ldots, n'_k)$ is a maximum m-bound allocation, if for any other m-bound allocation $(n''_1, \ldots, n''_k)$, $n''_i < n'_i$, $i = 1, \ldots, k$. With a slight abuse of notation, we assume that $(0, 0, \ldots, 0)$ is always an m-bound allocation that is subject to any sequence of weights W.

For example, suppose we have $N = (3, 8, 6, 9)$, $W = (1, 2, 2, 1)$ and $m = 15$. Then $(2, 4, 4, 2)$ is the maximum m-bound allocation of N subject to W. If $m = 5$, then the only m-bound allocation of N subject to W is $(0, 0, 0, 0)$.

To minimize the sum of ranges of generalizations, it is preferable to generalize a sensitive attribute value to a more specific subdomain (i.e., one far away from the root in the privacy-annotated hierarchy). Meanwhile, to make the tuples in the group follow the target distribution, the number of sensitive values generalized to a certain level of subdomains should preserve the ratio between the weights of those subdomains in the privacy-annotated hierarchy. This explains the intuition behind maximum allocation.

Algorithm 1 shows an optimal algorithm to the above problem. The algorithm is essentially a greedy algorithm. We start at the root node, and determine how many tuples have to be generalized to the root node. The basic idea is that we should push as many tuples as possible to the subtrees rooted at its children, as long as the distribution P is preserved (i.e., subject to the weights of its children). In detail, suppose we have n elements in g and the root node has k children $\mathcal{D}_1, \ldots \mathcal{D}_k$ in the domain generalization hierarchy. We determine what is the maximum number of tuples that should be generalized to subdomains in the subtree rooted at each $\mathcal{D}_i$, $i = 1, \ldots, k$, subject to the target distribution P. The remaining tuples must be generalized to the root domain.

EXAMPLE 1. *Here we use an example to show in detail how the algorithm works. Consider the privacy-annotated hierarchy H in Figure 3 and a multiset g of sensitive attribute values {30K, 30K, 40K, 40K, 50K, 60K}.*

Initially, we set m to $|g| = 6$, and call MinSumOfRange(g, H, m). For the two children of the root of H, we have $|g_{\{30K,40K\}}| = 4$ and $|g_{\{50K,60K\}}| = 2$. Since the ratio between the weights of {30K, 40K} and {50K, 60K} is 1:1 in H, we can at most have two subdomains in the subtree rooted at {30K, 40K} in the final P-private generalization. Otherwise, the target distribution cannot be preserved. In other words, the maximum m-bound allocation is $(2, 2)$. And the algorithm outputs $6-2-2 = 2$ generalized domain corresponding to the root, which is $\{30K, 40K, 50K, 60K\}$. This means that any P-private generalization should contain at least two root domains.

We next call MinSumOfRange(g, H_i, 2), where H_i is the subtree rooted at $\{30K, 40K\}$ and $\{50K, 60K\}$ respectively. For the left child, we have $g_{\{30K\}} = 2$ and $g_{\{40K\}} = 2$. The maximum 2-bound allocation of g would be $(1, 1)$. Therefore, at this step, we do not output any generalized subdomain $\{30K, 40K\}$ as $2 - 1 - 1 = 0$. A similar process happens for the subtree rooted at $\{50K, 60K\}$.

[3] This problem setting can be easily extended to handle categorical data with hierarchies, by defining the range of each subdomain in the hierarchy as the number of categorical values in that subdomain.

The algorithm continues to the leaf nodes in the hierarchy. The optimal generalization of g would be $\{\{30K\}, \{40K\}, \{50K\}, \{60K\}, \{30K, 40K, 50K, 60K\}, \{30K, 40K, 50K, 60K\}\}$.

Note that as we continue to the children of a node by calling function MinSumOfRange, we pass the subtree rooted at a child node the new bound. But we still pass g, all the tuples in the group, to the function. In other words, our algorithm does not need to decide which tuple's sensitive attribute value is generalized to which subdomain. This decision is not relevant because we will later permute all the generalized subdomains anyway. We only need to know what generalized subdomains are in an optimal solution. This is the essential reason for the efficiency of the algorithm.

Algorithm 1 MinSumOfRange(g, H, m): optimal generalization for the minimum sum of range problem

```
Let $\mathcal{D}$ be the root of $H$
Let $\mathcal{D}_1, \ldots, \mathcal{D}_k$ be the children of the root
Let $W = (w_1, \ldots, w_k)$ be the weights of $\mathcal{D}_1, \ldots, \mathcal{D}_k$ in $H$
Let $N = (|g_{\mathcal{D}_1}|, \ldots, |g_{\mathcal{D}_k}|)$
Compute $(n_1, \ldots, n_k)$, the maximum $m$-bound allocation of $N$ subject to $W$
Output $m - \sum_{i=1,\ldots,k} n_i$ copies of $\mathcal{D}$ to the final generalization $G$
for each subtree $H_i$ rooted at $\mathcal{D}_i$ do
  if $n_i > 0$ then
    MinSumOfRange($g$, $H_i$, $n_i$)
  end if
end for
```

We can compute $|g_{\mathcal{D}'}|$ for all subdomains $\mathcal{D}'$ bottom-up from all the leaf nodes. Therefore the complexity of this step is $O(|g|)$ assuming $|g|$ is greater than the number of subdomains in H. The above algorithm simply does a depth-first traversal of H, with complexity of $O(|H|)$. Thus the overall complexity is $O(|g|)$, i.e., linear to the size of the microdata.

THEOREM 1. *The multiset of subdomains generated by the algorithm MinSumOfRange is a P-private generalization with the minimum sum of ranges of subdomains. The complexity of MinSumOfRange is linear to the size of the microdata.*

4.2 Partitioning Quasi-Identifiers

We have shown an optimal algorithm that finds a P-private generalization with the minimum sum of ranges of subdomains for a given set of sensitive attribute values. Meanwhile, we observe that the more tuples in a group an aggregate query selects, the more accurate the bounds tend to be. For example, consider group 1 in Figure 2. If a query asks for the average salary of employees who live in the area with zipcode 911**, since all the selected tuples are in group 1, we can get very accurate bounds for this query. On the other hand, if the selected tuples are scattered in three different groups, then we can only get the trivial bounds [30K-60K].

Since a majority of aggregate queries involve range conditions, tuples with similar quasi-identifiers tend to be selected together. This observation suggests that, we may further partition the group specified by the microdata owner, and have tuples with similar quasi-identifiers together to improve query answering accuracy.

One possible approach is to preserve the optimal P-private generalization of the whole group. In other words, we further partition the tuples in the group into multiple subgroups such that the union of the optimal P-private generalization of each subgroup is the same as that of the whole group. And the optimization goal is to minimize the sum or max of the distances among the quasi-identifiers of tuples in all the subgroups. Here several possible distance functions can be used. However, clustering problems involving multi-dimensional attributes are generally intractable.

In this paper, we specifically consider a pragmatic approach, where we first partition a group into multiple subgroups such that tuples in each subgroup have the same quasi-identifiers. Then for each subgroup we find its optimal P-private generalization. We call this algorithm the *QI-SA* algorithm. In contrast, we use *SA-only* to denote the approach that uses only the optimal sensitive attribute generalization to answer queries without further partitioning.

One advantage of QI-SA is that, given any aggregate query, either all or none of the tuples in a subgroup are selected by the query. On the other hand, though the union of each subgroup's optimal P-private generalization is still a P-private generalization for the whole group, it may not be optimal. Thus in general we cannot claim that one approach always yields more accurate query answers than the other. In our experiments, we will perform a detailed comparison of the two approaches with different data sets.

5. INTEGRATION OF FAKE VALUES

In this section, we consider the use of fake values to further improve query answering accuracy. As shown in section 2, adding fake values may reduce the level of generalization. However, if we do not limit the number of fake values, though the sum of ranges of generalizations can always be reduced to 0, it may not improve the bounds for queries. The sum of ranges of generalizations can serve as a reasonable heuristic only when we restrict the number of added fake values. Formally, we study the following problem.

PROBLEM 5.1. *Given a multiset g of sensitive attribute values, a privacy-annotated hierarchy H corresponding to a target distribution P over $\mathcal{D}$, and a threshold t, find a multiset f of no more than t fake values, such that $MinSumOfRange(g \cup f, H, |g \cup f|)$ ($minSOR(g \cup f, H)$ for short) is minimized.*

5.1 Challenges

For simplicity, for the rest of our discussion, we assume H is a binary hierarchy. As mentioned before, with appropriate privacy annotation, a binary hierarchy can represent any distribution P over a domain $\mathcal{D}$.

EXAMPLE 2. *Suppose $g = \{1, 3, 5\}$, $t = 2$, and H is the uniform distribution over $\{1, \ldots, 8\}$. The optimal solution is for f to have only one fake value (either 7 or 8). We can easily verify that any f with two fake values is suboptimal. This example suggests that it is not sufficient to only examine set f that contains exactly t fake tuples.*

In fact, we can also show that there exists fake value set f which is locally optimal but suboptimal globally. By locally optimal, we mean that for any f_1 and f_2, where $|f_1| = |f| - 1$ and $|f_2| = |f| + 1$, we have $minSOR(g \cup f_1, H) > minSOR(g \cup f, H)$ and $minSOR(g \cup f_2, H) > minSOR(g \cup f, H)$.

EXAMPLE 3. *Consider $g = \{1, 3, 5, 7\}$ and $t = 4$. We can see that $f = \{1, 5\}$ is a locally optimal solution. The optimal generalization for g ∪ f is {[1-2], [3-4], [5-6], [7-8], [1-4], [5-8]}, whose sum of ranges is 10. For any f_1 with size 1, the optimal generalization for g ∪ f_1 is {[1-2], [3-4], [5-6], [7-8], [1-8]}, whose sum of ranges is 11. Similarly, for any f_2 with size 3, the optimal generalization for g ∪ f_2 is {[1-2], [3-4], [5-6], [7-8], [1-4], [5-8], [1-8]}, whose sum of ranges is 17. Though f is locally optimal, it is not a globally optimal solution, which should be $f' = \{2, 4, 6, 8\}$, as $minSOR(g \cup f', H) = 0$.*

The above example shows that Problem 5.1 cannot be solved by sequentially scanning fake value sets from size 0 to size t and using local optimality as a stopping condition.

To further show the challenges in solving this problem, let us consider a simpler case. Suppose we only allow to add a single fake value, and the target distribution is uniform. Assume in g the number of values in the left subtree is one more than that in the right subtree. When adding a fake value, a natural conjecture is that it is optimal to add a value belonging to the right subtree. Intuitively, by doing so the number of values in the two subtrees are balanced and thus closer to the target distribution. If this conjecture were true, the problem can be solved in a straightforward top-down manner.

Unfortunately, it turns out that adding fake values to the subtree with fewer values in g is not always the best choice.

EXAMPLE 4. *Consider g=*$\{1, 1, 1, 2, 3, 4, 5, 6, 7, 9, 9, 11, 11, 13, 13, 15, 15\}$*, where the domain* $\mathcal{D} = \{1 \ldots 16\}$. *Here g has one more element in the subdomain* $[1-8]$ *than in* $[9-16]$. *If we add a fake value to the right subtree, i.e.,* $[9-16]$*, then the optimal generalization would be three copies of [1-8], two copies of [9-10], [11-12], [13-14] and [15-16] respectively, and a single copy of [1-2], [3-4], [5-6], [7-8], [1-4], [5-8] and [9-16] respectively, whose sum of ranges is 46.*

Instead, if we add 8 which belongs to the left subtree, the optimal generalization would be two copies of [1-16], [9-10], [11-12], [13-14] and [15-16] respectively, and a single copy of [1-1], [2-2], ..., [8-8], whose sum of ranges is $38 < 46$.

The above example shows that when determining where to put fake values, we have to take the whole data set into consideration. Heuristics that only examine local properties (e.g., the balance of the subtrees at some level of the generalization hierarchy) would not yield optimal solutions.

5.2 Optimal Algorithm

Next we present an optimal algorithm for Problem 5.1. The basic idea is illustrated as follows. For simplicity, assume the target distribution represented by H is uniform. Let the two children of the root domain be $\mathcal{D}_1$ and $\mathcal{D}_2$. Suppose we want to find a set f of exactly k fake values, among which x comes from $\mathcal{D}_1$ and $k-x$ comes from $\mathcal{D}_2$, such that $minSOR(g \cup f, H)$ is minimized. Once x is fixed, we would know exactly how many values should be generalized to the root domain and from which side. In detail, let $n = |g_{\mathcal{D}_1}| + x$ and $m = |g_{\mathcal{D}_2}| + k - x$ (which are the numbers of values in $g \cup f$ that belong to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively). If $n \geq m$, then exactly $n-m$ values should be generalized to the root domain, and they are all from $\mathcal{D}_1$. Thus, to minimize $minSOR(g \cup f, H)$, we need to do the following. First, choose a set f_2 of $k-x$ fake values, such that $minSOR(g_{\mathcal{D}_2} \cup f_2, H_{\mathcal{D}_2})$ is minimized, where $H_{\mathcal{D}_2}$ is the hierarchy rooted at $\mathcal{D}_2$. Second, choose a set f_1 of x fake values, and then choose $f' \subset g_{\mathcal{D}_1} \cup f_1$, where $|f'| = n-m$, such that $minSOR(g_{\mathcal{D}_1} \cup f_1 - f', H_{\mathcal{D}_1})$ is minimized. In other words, for the side of $\mathcal{D}_1$, we need to insert x fake values and then remove $n-m$ values so that the resulting data set offers the minimum sum of ranges of generalizations. Note that this is not the same as simply inserting $x-(n-m)$ fake values, as the $n-m$ removed values may come from both the inserted fake values and the original data set $g_{\mathcal{D}_1}$.

In general, if H is not uniform, for both subdomains we may need to insert some fake values and then remove several others. For example, suppose in H the ratio between $\mathcal{D}_1$ and $\mathcal{D}_2$ is 2:1. Originally $|g_{\mathcal{D}_1}| = 10$ and $|g_{\mathcal{D}_2}| = 7$. Now assume we want to add 3 fake values to each of $g_{\mathcal{D}_1}$ and $g_{\mathcal{D}_2}$. In this case, we need to have 5 values generalized to the root domain, 1 from those in $\mathcal{D}_1$ and 4 from $\mathcal{D}_2$. That means, for $g_{\mathcal{D}_1}$, we need to insert 3 fake values and then remove 1 value. And for $g_{\mathcal{D}_2}$, we should insert 3 fake values and then remove 4 values. Here we see that sometimes the number of removed values may be more than that of those inserted.

The above discussion shows that Problem 5.1 is a special case of a more general problem: adding a set of k fake values to g and then removing r values, what is the minimum sum of ranges for the remaining data set after it is generalized and permuted to achieve P-privacy? We denote the problem as $minSORFake(g, k, r, H)$. Our discussion also suggests that this problem has the optimal substructure property, and is amenable to dynamic programming solutions. Specifically, let the ratio between $\mathcal{D}_1$ and $\mathcal{D}_2$ be $a : b$. Suppose among the k added fake values, x belongs to subdomain $\mathcal{D}_1$ and $k-x$ belongs to subdomain $\mathcal{D}_2$. Clearly, to minimize the sum of ranges of generalizations, the fewer values generalized to the root domain, the better. Let (u, v) be the maximum pair such that $u \leq |g_{\mathcal{D}_1}| + x$, $v \leq |g_{\mathcal{D}_2}| + k - x$ and $u : v = a : b$. Therefore, among the $|g| + k$ values, there should be at least $|g| + k - (u+v)$ to be generalized to $\mathcal{D}$.

Remember that we also need to remove r values. If $r \leq |g| + k - (u+v)$, we can simply take r values out from those generalized to the root domain. Otherwise, we have to further take values from $g_{\mathcal{D}_1}$ and $g_{\mathcal{D}_2}$. Thus, in general, we should let (u, v) be the maximum pair such that $u : v = a : b$, $u \leq |g_{\mathcal{D}_1}| + x$, $v \leq |g_{\mathcal{D}_2}| + k - x$, and $|g| + k - (u+v) \geq r$.

Once (u, v) is determined, we essentially break the solution to $minSORFake(g, k, r, P)$ into three parts: (1) those values generalized to the root domain $\mathcal{D}$; (2) those values generalized to the subdomains within the hierarchy rooted at $\mathcal{D}_1$; and (3) those values generalized to the subdomains within the hierarchy rooted at $\mathcal{D}_2$.

For the subdomain $\mathcal{D}_1$ and $g_{\mathcal{D}_1}$, we essentially insert x fake values and then remove $r_1 = |g_{\mathcal{D}_1}| + x - u$ values, i.e., it is the subproblem $minSORFake(g_{\mathcal{D}_1}, x, r_1, H_{\mathcal{D}_1})$. Similarly, for the subdomain $\mathcal{D}_2$, we have subproblem $minSORFake(g_{\mathcal{D}_2}, x, r_2, P_{\mathcal{D}_2})$, where $r_2 = |g_{\mathcal{D}_2}| + (k-x) - v$. In summary, we have

$$\begin{gathered} minSORFake(g, k, r, P) = \\ \sum_{0 \leq x \leq k} minSORFake(g_{\mathcal{D}_1}, x, r_1, P_{\mathcal{D}_1}) + \\ minSORFake(g_{\mathcal{D}_2}, x, r_2, P_{\mathcal{D}_2}) + (|g| - u - v - r)Range(\mathcal{D}) \end{gathered}$$

where u, v, r_1 and r_2 are calculated as described above. Algorithm 2 shows the pseudocode for the optimal algorithm. For simplicity, we present the algorithm using recursion, which can be easily converted to a standard dynamic programming algorithm.

To solve Problem 5.1 completely, we invoke minSORFake(g, k, 0, H) for $k = 0, \ldots, t$ in order to identify the optimal number of fake values to add. Another interesting observation is that, if we set $k = 0$, i.e., we do not insert any fake values, then algorithm *minSORFake* is exactly the same as algorithm *MinSumOfRange*.

EXAMPLE 5. *Let* $g = \{1, 2, 3, 5, 6, 7, 7\}$ *and* $\mathcal{D} = \{1, \ldots, 8\}$. *Assume H is a balanced binary tree that represents the uniform distribution over* $\mathcal{D}$. *Suppose we want to insert exactly 2 fake values into g, and thus invoke minSORFake(g, 2, 0, H). At the root level, we first let* $x = 0$ *and* $k - x = 2$*, i.e., all the fake values are from* $[5-8]$*, and will have* $sizeL = 3$ *and* $sizeR = 6$. *Since now the right subtree has 3 more values than the left one, we have to extract 3 values from the right tree and generalize them to the root domain. This is exactly what the algorithm does, as we set* $u = v = 3$, $r_1 = 0$ *and* $r_2 = 3$*, and invoke minSORFake(*$g_{[1-4]}$*, 0, 0,* $H_{[1-4]}$*) and minSORFake(*$g_{[5-8]}$*, 2, 3,*$H_{[5-8]}$*). Because we do not need to add or remove any values from* $g_{[1-4]}$*, minSORFake(*$g_{[1-4]}$*, 0, 0,* $H_{[1-4]}$*) is the same as MinSumOfRange(*$g_{[1-4]}$*,* $H_{[1-4]}$*,* $|g_{[1-4]}|$*).*

Algorithm 2 minSORFake(g, k, r, H): the minimum sum of ranges of generalizations when allowing fake values

```
Let $\mathcal{D}$ be the root of $H$
Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the left and right children of the root
if minSORFake($g$, $k$, $r$, $H$) calculated before then
    return the calculated result
end if
if $H$ is a leaf node then
    return 0
end if
$min$=inf;
for $x$=0 to $k$ do
    let $a : b$ be the ratio between left and right children of the root
    $sizeL$=$|g_{\mathcal{D}_1}|$+$x$
    $sizeR$=$|g_{\mathcal{D}_2}|$+$k - x$
    let $(u, v)$ be the maximum pair such that $u : v = a : b$, $u \leq |g_{\mathcal{D}_1}| + x, v \leq |g_{\mathcal{D}_2}| + k - x$, and $|g| + k - (u+v) \geq r$
    $r_1 = |g_{\mathcal{D}_1}| + x - u$
    $r_2 = |g_{\mathcal{D}_2}| + k - x - v$
    $tmp$ = minSORFake($g_{\mathcal{D}_1}$, $x$, $r_1$,$H_{\mathcal{D}_1}$)+ minSORFake($g_{\mathcal{D}_2}$, $k - x$, $r_2$,$H_{\mathcal{D}_2}$)+ $(|g| - u - v - r) * Range(\mathcal{D})$
    if $tmp < min$ then
        $min = tmp$
    end if
end for
return $min$
```

After minSORFake($g_{[5-8]}$, 2, 3,$H_{[5-8]}$) is invoked, we also first check the case $x = 0$. *As we need to eventually remove 3 values from the domain* $[5 - 8]$, *we will have* $u = v = 1$, $r_1 = 1$ *and* $r_2 = 3$. *Here we take a total of 4 values from the domains* $[5 - 6]$ *and* $[7 - 8]$, *among which 3 are removed (they are in fact those taken to the higher level to be generalized), and the remaining one is generalized to* $[5 - 8]$. *The algorithm will then continue to the next level of H.*

From the example, we see that the algorithm does not really need to generate specific fake values to add. Instead, it directly computes domain generalizations supposing optimal fake values are added.

5.3 Complexity Analysis

Suppose the hierarchy H is a balanced binary tree, and the domain of the sensitive attribute is $\mathcal{D}$ (i.e., the root domain of H). Observe that there are at most $2|\mathcal{D}|$ nodes in H. For each node, the algorithm can be invoked with at most $t + 1$ possible values of the number of fake values to be inserted in the node's subtree. And for each node and each possible number of fake values to insert in the node's subtree, there are at most $|g| + t$ values for the number of values to remove from the subtree. Thus a natural but loose bound for the number of cells in the dynamic programming matrix is $2(t + 1)(|g| + t)|\mathcal{D}|$. For each cell, the computation takes the minimum over $t + 1$ possible cells, which gives the time bound as $O(t^2(|g| + t)|\mathcal{D}|)$.

However, observe that, given g and H, even though algorithm *minSORFake* has two additional parameters k and r, r is not a free parameter. From the pseudocode we see that, once the number of fake values for each subdomains of the root is fixed (i.e., x and $k - x$), r is also uniquely determined. Let d be the depth of a node $\mathcal{D}'$ in H (where the root's depth is 0). Then the number of possible choices of fake values to add is bounded by $(t + 1)^d$, which, based on the above observation, also bounds the number of possible choices of values to remove from the subtree. Further, as the number of values to remove from a subtree cannot exceed the number of values in $\mathcal{D}'$, a tighter bound is given by $min((t + 1)^d, |g_{\mathcal{D}'}| + t)$.

Summed over all the 2^d nodes in the same level as $\mathcal{D}'$, we get the bound for the number of values to remove from all nodes in that level as $min((2t + 2)^d, |g| + t2^d)$. Note that the first term is small close to the root of the tree and gets much larger than the second term close to the leaves of the tree, if $\mathcal{D}$ is reasonably large.

Summed over all levels (and hence all the nodes) in the tree, the sum of the second term is given by $|g|(1 + log|\mathcal{D}|)$+ $2t|\mathcal{D}|$. Thus, even ignoring the first term in $min((2t + 2)^d, |g| + t2^d)$, this would yield a bound for the number of cells to be maintained in the dynamic programming matrix as $(t + 1)(|g|(1 + log|\mathcal{D}|)$+ $2t|\mathcal{D}|)$, which is much tighter than the bound $2(t + 1)(|g| + t)|\mathcal{D}|$ given above. Similarly, a much tighter bound for time complexity is given by $O(t^2(|g|log|\mathcal{D}| + 2t|\mathcal{D}|))$.

THEOREM 2. *A P-private generalization with minimum sum of ranges of subdomains when adding no more than t fake values is obtained using* $t + 1$ *invocations of Algorithm minSORFake. The space and time complexities of the algorithm are* $O(t(|g|log|\mathcal{D}| + t|\mathcal{D}|))$ *and* $O(t^2(|g|log|\mathcal{D}| + t|\mathcal{D}|))$ *respectively.*

6. EXPERIMENTS

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 commonly used in previous work on microdata anonymization [5,7]. We choose the same eight attributes as quasi-identifiers in our experiments in accordance to previous works. Since our approach focuses on numerical sensitive attributes, 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 synthetic data sets with the same schema as the Adult Database. We populate a table with different numbers of tuples, assuming certain distributions of the capital loss attribute. We also consider the correlation between "capital loss" and quasi-identifiers. The details of the synthetic data sets will be described later when we present the experimental results.

We design the following experiments to verify the effectiveness of our approach. In all the experiments, we evaluate the accuracy of the bounds derived from an anonymized table. Specifically, let l and u be the lower and upper bounds of an aggregate query result r. We define $err = (u - l)/r$ to be the relative error of the bounds. The smaller err is, the more accurate the bounds are.

The performance of anonymization through SA generalization. We show the accuracy of aggregate query answers of different query sizes, when applying the SA-only algorithm and the QI-SA algorithm on both real and synthetic data sets.

The impact of adding fake values. We show the impact of integrating different number of fake values with the QI-SA and SA-only schemes.

The impact of the target distributions. We study the accuracy tradeoff when the data owner specifies different target distributions other than the actual distribution of the sensitive attributes in the original microdata.

Enforcing different target distributions on different parts of the data. We show the effectiveness of our algorithm when we enforce different target distributions for different parts of the data.

Next, we describe each set of experiments in detail. Unless otherwise specified, the domain generalization hierarchy used in these experiments is a balanced binary tree.

6.1 Performance of Anonymization through Sensitive Attribute Generalization

We first study the relative errors of our algorithms applied on different data sets. In all experiments in this section, the target distribution is set as the source distribution of sensitive attribute values in the overall microdata table (like t-closeness). 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 $[X, Y]$. More specifically, given a range R, we randomly pick 100 pairs of X and Y from the domain of the age attribute such that $Y - X = R$. We then report the average relative error of these 100 queries.

Figure 5 shows the experimental results when using SA-only and QI-SA respectively. We observe that, as the total number of tuples selected increases, the relative error introduced by SA-only drops dramatically, while that introduced by QI-SA does not change much. This is because for SA-only, when the query range increases, the number of tuples touched in each group also increases, which results in a more accurate estimate. For QI-SA, each "age" value is by itself a group, and tuples in each group are either completely touched or not touched at all by a query. So the derived bounds do not change much when the query range changes.

Another observation is that when the query range is small, SA-only gives quite inaccurate bounds. The reason is that with small query range the selected tuples only account for a very small portion of each group, which results in bounds with poor accuracy. When the query range increases, as SA-only minimizes the sum of ranges across all generalized subdomains, it produces bounds with better accuracy than QI-SA.

Meanwhile, as QI-SA generates many more groups than SA-only, each group is of a much smaller size. Thus when using QI-SA, the sensitive attribute values in many groups are generalized to the root domain, because the distribution of each group with the same quasi-identifiers is often quite different from the target distribution. So even when the query range increases, the accuracy of the bounds does not improve.

To further validate our evaluation, we compare QI-SA and SA-only on two synthetic data sets, with different sensitive attribute distributions and correlations between quasi-identifiers and the sensitive attribute. In the first data set, there is no correlation between quasi-identifiers and the sensitive attribute. We randomly generate 10k tuples whose sensitive attribute values for each tuple follows a normal distribution. The only quasi-identifier "age" is randomly generated uniformly in the range $[17, 79]$.

The experimental results are shown in Figure 5. We see that for this data set QI-SA consistently produces accurate bounds. This is because the sensitive attribute values in each age group follow the overall distribution closely. Thus, only a few generalizations are needed to achieve the privacy goal. As the query range increases, more tuples are selected, and the relative errors of SA-only quickly drops. In particular, when most of the tuples are selected by a query, SA-only outperforms QI-SA, due to its emphasis on a global optimal subdomain generalization.

Intuitively, if a strong correlation exists between quasi-identifiers and the sensitive attribute, tuples in each age group generated by the QI-SA algorithm tend to have similar sensitive attribute values, which need to be generalized to higher subdomains to achieve the privacy goal. Our next experiment aims to investigate the impact of correlation on query answering accuracy. We compare QI-SA and SA-only when varying the correlation between quasi-identifiers and the sensitive attribute. In the second synthetic data set, we introduce a correlation between "age" and "capital loss". First, "age" is generated following a uniform distribution in the range $[17, 79]$. We assume a positive correlation between "age" and "capital loss". For each tuple with age a, we generate its capital loss following a normal distribution with mean v. The higher one's age is, the larger v we choose. The size of the data set is also 10k.

Figure 5 shows the experimental results. We see that SA-only consistently outperforms QI-SA. In fact, the trend of SA-only is similar to the case with no correlation between "age" and "capital loss". This suggests that an optimal P-private generalization of the whole group can accommodate such correlation well. On the other hand, QI-SA, as expected, performs poorly, since it only considers the sensitive attribute values inside a single subgroup, whose distribution is very different from the overall distribution.

Overall, the above experiments suggest that our optimization algorithms are capable of supporting aggregate query answering with reasonable accuracy while protecting privacy. In particular, SA-only tends to answer aggregate queries more accurately if quasi-identifiers and the sensitive attribute are correlated. Otherwise, QI-SA offers more accurate query answering for smaller query ranges.

6.2 Impact of Integrating Fake Values

We study the impact of integrating fake values with the QI-SA and SA-only schemes. We first look at QI-SA. In each QI partition of size N, we allow up to $K = x\% \cdot N$ fake values. Different number of fake values allowed are denoted "$+x\%$" in the legend. Figure 5 shows the impact in the real data set. We see that adding a few fake tuples helps reduce the error rates effectively. But when adding too many fake tuples (20%), the uncertainty introduced by fake tuples begins to dominate, resulting in worse query accuracy. Figure 5 shows the impact in the non-correlated synthetic data. As explained before, SA values in each partition follows the target distribution closely. Thus very few generalizations are needed initially. When adding a few fake tuples (5%), it helps to reduce error rates. Adding more fake tuples reduces query accuracy due the same reason described above. Figure 5 shows the impact on the correlated synthetic data. Since there are many generalizations initially in this case, adding fake tuples keeps reducing error rates.

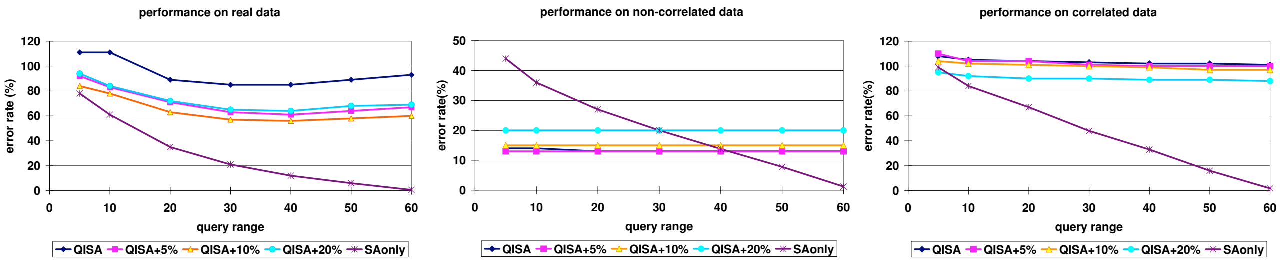
Next we look at the SA-only scheme. To study the impact of adding fake values, we randomly remove 10% of the source data to make its distribution different from the target distribution.[4] Figure 6 shows the results, from which we have the same observation as from the QI-SA case.

6.3 Impact of Target Distributions

We investigate the relative errors of QI-SA and SA-only when we have different target distributions and domain generalization hierarchies. As shown above, QI-SA does not perform well when there is a strong correlation between quasi-identifiers and the sensitive attribute. Thus, in this experiment we use a data set in which quasi-identifiers and the sensitive attribute are not correlated.

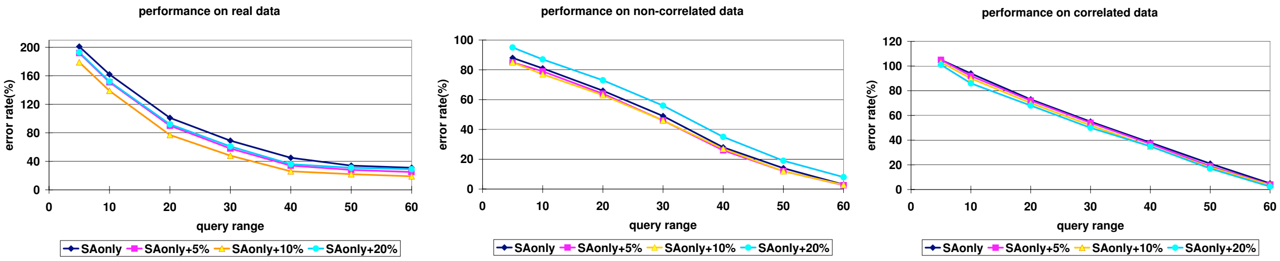
The first experiment studies the trends when the target distribution is different from the actual distribution. In the synthetic data set of size 10k, the sensitive attribute "capital loss" is generated following a uniform distribution in the range $D = [1024, 3072]$. The target distribution is set to be a normal distribution. We change the variance of the target distribution by adjusting a parameter r, where the variance of the normal distribution equals $range(D)/r$. The bigger r, the smaller the variance, and the larger the difference between the target and the actual distributions. We examine the relative errors of range queries whose ranges are set to be 50. The

[4] When the target distribution is the same as the source distribution, the SA-only scheme is the same as global permutation, and no generalization is involved. So adding fake values has no effect.



(a) Real data (b) Non-correlated data (c) Correlated data

Figure 5: Relative error rate of QI-SA and SA-only, and QI-SA with fake values



(a) Real data (b) Non-correlated data (c) Correlated data

Figure 6: Relative errors of SA-only on both the real and the synthetic data sets

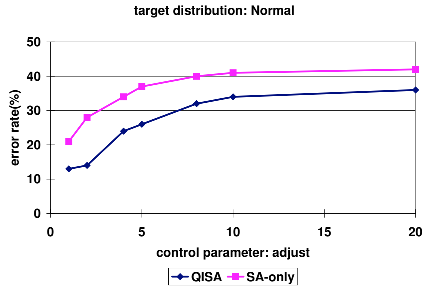


Figure 7: Relative errors of QI-SA and SA-only when target distributions are normal distributions with different variances

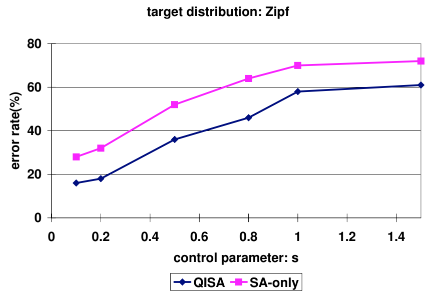


Figure 8: Relative errors of QI-SA and SA-only when target distributions are Zipf distributions with different parameters

experimental result is shown in Figure 7.

We see that when the target distribution is close to the actual uniform distribution, both algorithms offer better accuracy. As r increases, the error rates also increase. This is expected as more tuples' sensitive attribute values have to be generalized to higher domains to make them follow the target distribution that is far different from the actual distribution.

We also investigate the impact when the sensitive attribute values in the original microdata are skewed. In particular, we set the actual distribution of the sensitive attribute values to be a zipf distribution. All values in the sensitive attribute domain are ordered and assigned a rank correspondingly. Denote the size of the domain as N. For any element with rank k, its probability is set to be $P(k) = \frac{(1/k)^s}{\sum_{i=1}^{N}(1/i)^s}$, where s is a control parameter. The smaller the value s is, the less skewed the data are. In our experiment the control parameter s is set to 0.1. The target distribution is also a zipf distribution, with s set to different values. The results are shown in Figure 8. We see that when the target distribution has parameter $s = 0.1$, we have the best accuracy for query answering, as it is the same as the actual distribution. As s increases, the difference between the target distribution and the actual distribution is enlarged, which results in increased relative errors.

We also conduct the same experiment on a skewed data set with a strong correlation between the quasi-identifiers and the sensitive attribute. Specifically, the sensitive attribute values are generated following the Zipf distribution as described above. Meanwhile, the quasi-identifier of a tuple is generated to be positively correlated with its sensitive attribute. Figure 9 shows the experiment results. We see a similar observation to that of Figure 5.

Our scheme does not impose any constraints on the structure

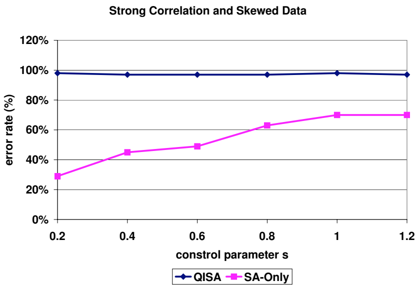


Figure 9: Relative errors of QI-SA and SA-only when target distributions are Zipf distributions and the QI and SA are correlated

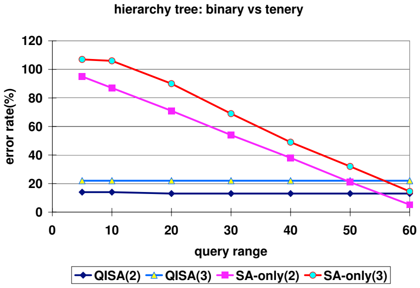


Figure 10: Relative errors of QI-SA and SA-only when using different domain hierarchies

of the domain generalization hierarchy The data owner has complete freedom to design their own domain hierarchy. Although it is impossible to enumerate all possible hierarchies, there are some general guidelines for choosing domain hierarchies. In the following experiment, we choose a balanced binary tree and a balanced ternary tree as two domain hierarchies. In the legend of Figure 10, we use (2) and (3) to represent them respectively. In the data set 10k sensitive attribute values are uniformly generated in the range [1024,3072]. The target distribution is also set to be a uniform distribution for simplicity.

We see that the case with the binary hierarchy results in lower relative errors. This is because it has more subdomains than in the ternary one. Thus statistically we have more choices of smaller subdomains when generating optimal generalizations, introducing less uncertainty to the microdata.

6.4 Anonymizing Different Parts of Data

One feature of our scheme is to accommodate different target distributions for different parts of the data. Next we study how it affects the quality of query answering. We first conduct experiments on the real data set. We study two cases where we partition the data into 2 groups and 10 groups respectively, based on the "age" attribute. In each case, the target distribution of each part is set to be the distribution of the source data in that group. In the legend of Figure 11, SAonly-k means we partition the table into k groups, and in each group we apply the SA-only algorithm to generalize the sensitive attribute values. QISA-k is interpreted similarly.

We see that the accuracy of both SA-only and QI-SA improves as the number of partitions increases. The reason is that as the number of groups increases, each group has fewer tuples, and the same range query will cover a bigger portion of each group, which leads to better accuracy of query answering.

We conduct a similar experiment on the correlated synthetic data

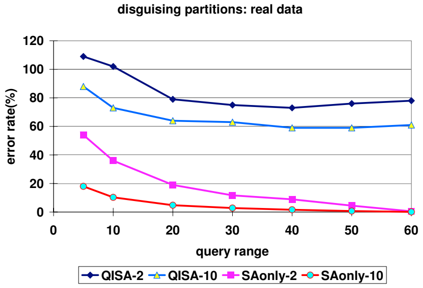


Figure 11: Accommodating target distributions of different parts of the real data set

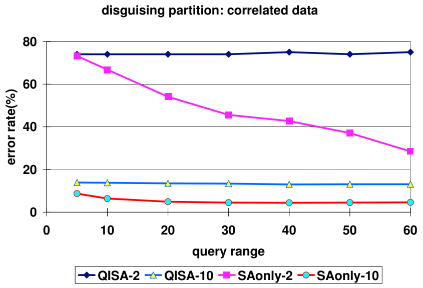


Figure 12: Accommodating distribution of different parts of a synthetic data set

set described in Section 6.1. In this experiment we use a uniform distribution as the target distribution for each group. The range of each corresponding uniform distribution is the actual range of the sensitive attribute values in that group. The results are shown in Figure 12. As with real data sets, we observe that as the number of groups increases, the relative errors of both SA-only and QI-SA decreases. When we have more groups, the ratio of tuples touched by a range query in each group increases. Another reason contributing to this result is the strong correlation between "age" and "capital loss". As we split the data based on "age", the range of sensitive attribute values in each group also becomes smaller. Thus we can use a smaller hierarchy tree for each group. This leads to smaller generalizations, and reduces the error further.

Overall, the above experiments show that by accommodating different target distributions, we can have a finer protection of microdata. Further, the generalizations produced by QI-SA and SA-only allow aggregate query answering with reasonable accuracy.

6.5 Algorithm Complexity

Figure 13 shows the running time of algorithms QI-SA and SA-only when varying the number of tuples in the data set. As expected, it is linear to the size of a data set.

7. RELATED WORK

The privacy vulnerability of de-identified microdata was first discussed by Sweeney [10], who showed that medical records of many individuals could be uniquely identified, after linking a de-identified medical database with voter registration records. Sweeney further proposed k-anonymity as a model for protecting privacy of microdata, and introduced quasi-identifier generalization and record suppression as two techniques to achieve k-anonymity [11].

In [9], Samarati presented a framework of quasi-identifier generalization and suppression to achieve k-anonymity. Given pre-

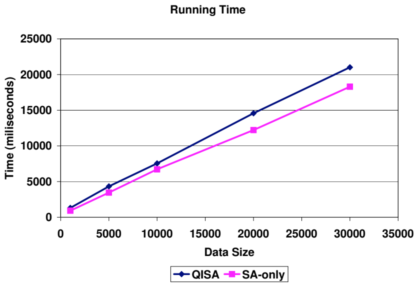


Figure 13: The time complexities of QI-SA and SA-only.

defined domain hierarchies of quasi-identifiers, the problem of k-anonymity is thus to find the minimal quasi-identifier generalization so that, for each tuple t in the released microdata table, there exist at least $k-1$ other tuples which have the same quasi-identifiers as t. Samarati also designed a binary search algorithm to identify minimal quasi-identifier domain generalizations.

The concept of ℓ-diversity was introduced by Machanavajjhala et al. in [7] to prevent attackers with background knowledge. In [6], distribution of sensitive attributes is first considered. Based on this, a more robust privacy measure (referred to as t-closeness) is proposed. Their work guarantees that the distribution of any sensitive attribute within each group (equivalence class) is close to its global distribution in the table. In [8], Martin et al. develop a language to describe public background knowledge and design an algorithm to measure the worst-case information disclosure for different anonymization techniques. Brickell and Shmatikov [1] analyze the privacy/utility tradeoff in microdata anonymization, and showed that quasi-identifier generalization often incurs significant negative impact on data utility.

All the above works focus on introducing less imprecise information to microdata. But they did not discuss their impact on the accuracy of aggregate queries. Xiao et al. [13] propose to achieve k-anonymity by separating quasi-identifiers and sensitive attributes into two tables. These two tables are 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. They show that when quasi-identifiers are maintained, the accuracy of aggregate reasoning is improved a lot, as the probability of each tuple being touched is known. As with most other works discussed above, however, this work only focuses on categorical sensitive attributes. Their techniques cannot be directly applied to handle numerical sensitive attributes, which was the focus of the work by Zhang et al. [15].

None of the above works can transform the microdata such that the distribution of the sensitive attribute of each tuple appears to follow a target distribution.

Recent advances in data anonymization go beyond tabular data to handle data with richer structures, including bipartite graphs [3] and social network graphs [4, 16]. Privacy models for each type of data and corresponding anonymization techniques are proposed, which are complementary to our approach. In particular, we observe that the basic idea of this paper can be similarly applied to the above data models as well.

Privacy models have also been proposed for the anonymization of microdata that need to be updated [2,14]. m-invariance is one of the representative models [14]. The basic idea is to keep unchanged the set of sensitive attribute values in the group that a tuple belongs to even though the tuple may be put into different groups in different versions of the microdata. The concept of distribution-based anonymization can also be extended for dynamic microdata.

8. CONCLUSION

In this paper, we proposed a novel technique for distribution-based microdata anonymization, which combines sensitive attribute generalization and permutation along with the use of fake values. The proposed scheme allows the microdata owner to flexibly decide the grouping of individual tuples while still protecting privacy. We further designed algorithms that produce optimal sensitive attribute generalization to improve the accuracy of ad hoc aggregate queries over the released microdata.

There are many interesting issues to be further explored. In particular, we would like to extend the techniques in this paper to support personalized privacy and other types of data. We are also interested in efficient algorithms to answer more complex data analysis, such as data mining, over anonymized microdata.

9. REFERENCES

[1] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, 2008.

[2] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *SDM*, 2006.

[3] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang. Anonymizing bipartite graph data using safe groupings. In *VLDB*, 2008.

[4] M. Hay, G. Miklau, D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. In *VLDB*, 2008.

[5] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Incognito - efficient full-domain k-anonymity. In *SIGMOD*, 2005.

[6] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: privacy beyond k-anonymity and ℓ-diversity. In *ICDE*, 2007.

[7] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: privacy beyond k-anonymity. In *ICDE*, 2006.

[8] D. J. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Y. Halpern. Worst-case background knowledge for privacy-preserving data publishing. In *ICDE*, 2007.

[9] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

[10] L. Sweeney. Guaranteeing anonymity when sharing medical data, the datafly system. *Journal of the American Medical Informatics Association*, pages 51–55, 1997.

[11] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

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

[13] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *VLDB*, 2006.

[14] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *SIGMOD*, 2007.

[15] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu. Aggregate query answering on anonymized tables. In *ICDE*, 2007.

[16] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In *ICDE*, 2008.