ABSTRACT

ZHANG, QING. Microdata Privacy Protection Through Permutation-Based Approaches. (Under the direction of Professor Ting Yu and Professor Munindar P. Singh).

Data analysts often prefer access to data in the form of original tuples(i.e., microdata), instead of pre-aggregated statistics, since the former offers advantages in information flexibility and availability. Two problems should be addressed before releasing microdata. First, individual's privacy needs to be adequately protected. In general, the data will be anonymized before sharing. Second, the utility of the anonymized microdata should be maintained and common aggregate queries should be answered with reasonable accuracy.

Most existing works on microdata anonymization are based on attribute generalization. Though popular, these approaches have limitations: the generalization of attributes make it difficult to answer typical aggregate queries with reasonable accuracy.

This dissertation investigates new techniques to address the limitations of existing approaches. We propose to anonymize microdata through permutation-based approaches. In particular, we first extend existing privacy goals to better fit the protection requirement of numerical data, and develop a scheme to achieve this privacy goal through sensitive attribute permutation. Second, we propose a stronger privacy goal where an attacker can only learn from the microdata that an individual's sensitive attribute follows a pre-specified target distribution, but nothing more. We combine sensitive attribute permutation and generalization techniques to achieve this goal. To get better query answers when the target distribution is far from that of the original microdata, we further provide mechanisms to allow users to better control the tradeoff between privacy and accuracy. Third, we extend our techniques to anonymize graph data and support the accurate answering of queries that involve graph properties. Specifically, we partition the nodes and relabel (a form of permutation) the nodes within the same partition. Finally, we study anonymization techniques that can support personalized privacy, which allows individuals to flexibly control the privacy protection they desire.

Microdata Privacy Protection Through Permutation-Based Approaches

by

Zhang, Qing

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research and Computer Science

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Peng Ning	Dr. Rada Chirkova
Dr. Munindar P. Singh Co-Chair of Advisory Committee	Dr. Ting Yu Chair of Advisory committee

DEDICATION

To my parents *Qingan Zhang and Jurong Yang*

my beloved wife *Qian Zheng*

and

my dear daughter *Elisa Zhang*

BIOGRAPHY

Qing Zhang received his B.S. degree in automatic control from University of Science and Technology of China in 2001. In spring 2002, he started his Ph.D. study in the Operations Research program in North Carolina State University. From spring 2004, he's a Ph.D. student co-major in Operations Research and Computer Science. During his graduate studies at NC State, he interned at AT&T Labs - Research. His research interests are in information security and privacy. In particular, he is interested in microdata privacy, reputation-based trust management, and sensor network security.

ACKNOWLEDGMENTS

This dissertation would not be possible without the support and help from many professors, friends, and my family members over many years.

First, I would like to thank my advisor Dr. Ting Yu. I feel very fortunate to have such a great advisor like you for my Ph.D study. Thank you for your patience, insightful suggestions, financial support and unending encouragement during my Ph.D research. Additionally, I would like to thank my co-advisor Dr. Munindar Singh, for your guidance and suggestions to my research. I would also like to thank my committee members Dr. Peng Ning and Dr. Rada Chirkova, for your valuable feedback and comments on my research.

I would like to thank Dr. Divesh Srivastava and Dr. Graham Cormode from AT&T - Labs Research, and Dr. Nick Koudas from University of Toronto. It has been an honor to work with you. And I have learned so much from our collaborations. The time I spent as a summer intern at AT&T labs in 2007 will always be a precious memory in my life.

I would thank all my friends in the Cyber Defense Laboratory, for their help to my research, and the happy times spent together in the office: Vinod Arjun, Keith Irwin, An Liu, Donggang Liu, Pai Peng, Kun Sun, Pan Wang, Dingbang Xu, Yan Zhai, Qinghua Zhang, Yi Zhang. And I would like to thank my friend from the Operation Research program, for their help when I initially when I came to the U.S., and the friendship that encourage me all the time during my study in NCSU: Gangshu Cai, Chuan Lin, Wei Wan, Meng Wang, Yong Wang, Xinmin Wu, Yufeng Xin, Xiaoqing Yang, Yipeng Yang, Zhe Zhang, Wei Zhang, Jie Zhong, Xiang Zhou.

Finally, I would like to give my special thanks to my parents Qingan Zhang and Jurong Yang, and my beloved wife Qian Zheng, for their continued support and encouragement to me. And also thank my dear daughter Elisa for the happiness you bring to my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Privacy-Preserving Data Sharing

With the fast development of computer hardware and software, and the rapid computerization of businesses and government operations, large amount of data have been collected. These data often need to be published, shared with, or outsourced to collaborating companies for further processing. For example, the government may need to publish the census data with household income information in a certain area; a hospital may need to share its patient records with public health researchers; a loan company may need to publish its customer finance data to demonstrate its business rules, and so on.

Such data often contains private information, and should not be disclosed directly. In the above example, the household income of a particular family in the census data, the health record of a particular patient, and the financial history of any individual in the loan report are all sensitive information, and its privacy should be maintained.

Traditionally, the data owner often choose some representative statistics to publish, or pre-aggregate parts of the data that others might be interested in. In this way, an individual's privacy is better protected. However, data published in these forms lack flexibility. Others can only learn about the pre-computed statistics, but nothing else.

In recent years, researchers have proposed to publish data in the form of microdata, i.e., data in the original form of individual tuples. Obviously the release of microdata offers significant advantages in terms of information availability, as the original records are kept and people can issue arbitrary queries they are interested in. So it is particularly suitable for ad hoc analysis.

However, the release of microdata raises privacy concerns when records containing

sensitive attributes (SA) of individuals are published. Existing privacy practice relies on de-identification, i.e., removing explicit identification information (e.g., name, SSN, home address and telephone numbers) from microdata. However, it has been well recognized [1, 2] that simple de-identification is not sufficient to protect an individual's privacy. One's other attributes (so-called *quasi-identifiers, or QI for short*, such as age, zip code, date of birth and race) are usually needed for data analysis, and thus are kept after de-identification. Individuals' sensitive information may often be revealed when microdata are linked with publicly available information through quasi-identifiers. A famous example is given by Sweeney in [2], where she successfully identified the governor of Massachusetts using only his date of birth, gender, and ZIP code from local hospital records, and then combine this information with the census database.

k-anonymity [2] is a privacy model to address the above privacy problem. Through domain generalization and record suppression, k-anonymity guarantees that publicly available information cannot be related with less than k records in a microdata database. In other words, given a sensitive attribute value in microdata, an attacker can at most relate it to a group of no fewer than k individuals instead of any specific one. The concept of ℓ-diversity [3] was recently proposed to further protect privacy in microdata. It is based on a stronger attack model where an attacker is assumed to have the knowledge that both the record corresponding to an individual and some values of its quasi-identifiers appear in a microdata database. In its basic scheme, each individual will be associated to at least l different sensitive attribute values in the anonymized data. Thus in order to recover the original mapping, the attacker needs at least $l - 1$ pieces of additional background information.

The privacy goals of k-anonymization and ℓ-diversity are suitable for categorical sensitive attributes, such as the disease attribute in a patient record table. They assume that different attribute values are incomparable. In ℓ-diversity, as long as it is ensured that an individual's sensitive attribute value can at most be narrowed down to a group of no less than k tuples with no less than ℓ distinct values, one's privacy is protected. In practice, besides categorical attributes, many sensitive attributes in microdata databases are in fact numerical data, e.g., one's salary, investment gains or losses. The privacy goals of k-anonymity and ℓ-diversity are often not sufficient to protect numerical attributes. For example, even when ℓ-diversity is satisfied, if the group of salary values falls into a narrow range, an attacker can still obtain sensitive financial information of an individual. Therefore, it is important to define new privacy goals for the protection of numerical sensitive attributes in microdata.

In some scenarios the distribution of specific attribute values in the released microdata

can be as sensitive as the private information of individual records. For example, consider a hospital in Beverly Hills that treats celebrities. Since reporters often keep monitoring which celebrities have checked into the hospital, it would be desirable for the hospital to disguise the distribution of the released sensitive attributes (e.g., symptoms and prescriptions) to a different "target distribution" which, e.g., exactly or approximately mirrors the distribution from the general population of patients in the Los Angeles area. It will be very useful to propose an anonymization scheme that ensures an attacker can only derive from the microdata that an individual record's sensitive attribute follows a pre-specified target distribution.

Most of existing works are focusing on the privacy problems of tabular data. We also observe that private data often arises in the form of associations between entities. An example is the products bought by customers at a pharmacy. The set of products being sold and their properties are public knowledge, and it may be no secret which customers visit a particular pharmacy. However, the association between a particular individual and a particular medication is often considered sensitive, since it is indicative of diseases or health issues that they have. The most natural way to model such data is to use a bipartite graph: nodes represent entities, and edges indicate associations between them. Graph data pose more privacy challenges than tabular data: a single piece of background information combined with the graph structure can often leak a large amount of information [4]. New anonymization schemes are needed which can address the challenging privacy problems associated with graph structures.

In this dissertation, we propose solutions to all the privacy problems mentioned above. Furthermore, we identify that since most existing approaches achieve privacy through generalization of quasi-identifiers [5, 6, 3, 7, 8, 9], they cannot answer aggregate queries with any reasonable accuracy, thereby reducing the utility of the released microdata. We emphasize that complementing the desire for privacy of microdata is the need to support ad hoc aggregate analysis that select subsets of records based on arbitrary conditions on the quasiidentifiers and compute aggregates over sensitive attributes (e.g., what is the average salary of men over age 50 in Texas?). For each of the privacy models we addressed above, we also investigate effective techniques to support accurate aggregate query answering on microdata while preserving privacy.

1.2 Scope of The Research

Privacy-preserving data sharing is a broad topic. In this dissertation, We restrict our discussion within a carefully chosen scope. Solutions within this scope can serve as a foundation for

more complex scenarios.

1.2.1 Data model

In this dissertation, we study the privacy of two types of data.

- First we consider the anonymization of a de-identified tabular data from one table, where each entry of the table corresponds to an individual. Attributes of each entry can be separated into quasi-identifiers and a sensitive attribute. For multiple tables, if they are non-correlated, which means there is no association between their attributes, we can anonymize them with our approach separately. If there are some associated attributes, we can construct a combined view on these tables based on those attributes. When we do need to publish multiple correlated tables separately, the problem is much harder. As pointed out by Yao et al. [10], the problem of checking k-anonymity in multiple views is generally NP-hard. We leave it as possible future work.

- For tabular data, we assume the data is already de-identified. The remaining attributes are either quasi identifiers or sensitive attributes. If there exist no quasi identifiers in the table, or if some of the quasi identifiers are also sensitive, it is difficult for attackers to link the identifiers in the outside database and tuples in the microdata. Thus we do not consider such situations.

- In Chapter 4, we will look at the privacy of bipartite graph data. We assume in the graph nodes represent entities, and edges indicate associations between them. Specifically, we only consider unweighted, and undirected graphs, and each pair of nodes have only a single edge between them.

1.2.2 Publishing model

We assume a static model of microdata release. In other words, we assume there is no change to the original data. Therefore, once an anonymized microdata table is published, there is no need to update it. Many real applications work in this fashion. For example, for streaming data, we can always collect data block by block and treat each block as an independent table. For the case of modification and deletion in the old data, the problem becomes more complicated, as attackers can track the difference in the anonymized tables before and after the modification. This will introduce

more privacy leakage. Currently there is no solution to this kind of situation yet. We also leave it to future work.

1.2.3 Query model

We consider queries that select subsets of records based on arbitrary conditions on the quasi identifiers, and computes aggregates on sensitive attributes. Such queries are most commonly used in OLAP. For graph data, we further define different types of queries, based on their conditions on the attributes, graph structure, or different combinations of them in chapter 4.

1.2.4 Privacy model

For privacy of tabular data, we focus on protecting the sensitive attribute of individuals. In other words, the goal is to prevent attackers from knowing the sensitive attribute values of individuals. For bipartite graph data, we focuses on the privacy of the association between the two types of entities. We do not constrain if attackers can infer whether someone or some entity is in the database. In other words, we do not take into account the "existential sensitivity", where the mere fact that there exists a record for a specific individual Alice in the microdata table may also be considered sensitive, even though Alice's sensitive attribute is unknown. The reason is, as stated in [3], besides public databases, attackers may often have external background knowledge. For example, Bob may physically see that Alice checked into a hospital. Thus, it is difficult if not impossible to prevent such information leakage. In this dissertation, revealing one's sensitive attribute values is considered a privacy violation, but revealing the existence of a record with specific quasi-identifiers is not.

1.2.5 Attack model

For tabular data, we assume the only information attackers can access is the published anonymized table, and some other publicly available database that contains the association between the unique identifiers and quasi-identifiers in the microdata. We do not consider insider attacks. In other words, we assume attackers cannot access the original data, and do not have priori knowledge of the correlation between quasi-identifiers and sensitive attributes.

For the bipartite graph data, we also assume that attackers can only access the original published data. They are *static*: they cannot inject nodes or edges to the graph. In some cases

attackers may have some information about the graph structure. It is more complicated and we will leave the detailed discussions to chapter 4.

1.3 Contributions

The goal of privacy protection in microdata is essentially to break the association between the identifiers in publicly available information and the sensitive attributes in microdata, due to quasi-identifiers. Existing approaches are mostly based on generalizations on quasi-identifiers (QI-generalization) and aim to weaken the link between quasi-identifiers in publicly available databases and microdata. However, QI-generalization based approaches cannot maintain much utility of the anonymized data. In this dissertation, we propose *permutation*-based approaches on sensitive attributes (SA-permutation) to reduce the association between quasi-identifiers and sensitive attributes in microdata. We also identify the privacy requirements for several specific scenarios, and design corresponding anonymization schemes based on permutation approaches. In particular, the contributions of this thesis include the following:

1. We propose permutation-based techniques on sensitive attributes to anonymize microdata table and protect privacy. We show that SA-permutation based anonymization techniques can always answer aggregate queries more accurately than QI-generalization based approaches.

2. We propose (k, e)-anonymity, a new privacy goal to better capture privacy protection for numerical sensitive attributes in chapter 2. Besides requiring a group of sensitive attribute values to have no less than k distinct values, the proposed privacy goal requires the range of the group to be larger than a certain threshold e. We design an anonymization scheme to achieve this privacy goal based on SA-permutation. We further optimize the scheme so that it can provide the best query answers to aggregate queries, and anonymization algorithm runs efficiently on large dataset.

3. We propose the distributional privacy model in chapter 3, which allows data owner to specify an arbitrary target distribution, and attackers can only derive from the microdata that any individual's sensitive attribute follows the target distribution but nothing else. In order to achieve this privacy goal, we propose a novel anonymization scheme which is also based on SA-permutation. In this scheme we first partition the SA values, generalize these values along a predefined hierarchy tree on the sensitive attribute domain. Then permutate the generalized

domains within each partition. We also develop efficient algorithms to optimize this scheme and provide the best query answers to aggregate queries.

When the target distribution is far from the source distribution, the query accuracy is often poor. We further propose two approaches to achieve better accuracy for query answering. The first approach allows data owners to approximate the target distribution in a controlled manner to tradeoff distributional privacy protection and accurate query answering during anonymization. The second approach is to inject fake tuples into the anonymized dataset.

4. Private data often comes in the form of association between entities. In chapter 4 we show that existing anonymization techniques for tabular data cannot give accurate answers to queries on graph properties: small changes on edges will change aggregate graph properties radically. We propose a novel group of anonymization techniques for bipartite graph data, called (k, l)-groupings. Graph data pose more security challenges than tabular data. A single piece of background information combined with the graph structure can often leak a large amount of information. We demonstrate such attacks, and study the security properties of (k, l)-groupings in detail.

5. In practice, users may not care about their privacy that much, which makes it possible that we relax the privacy protection for some users, and provide more accurate answers to aggregate queries. We initialize this privacy model in chapter 5 to give users choices to specify their own privacy preferences and proposed the personalized privacy model.

1.4 Dissertation Outline

In the rest of this dissertation, we describe different privacy goals, as well as their respective solutions in each chapter. Specifically, we will describe the (k, e)-anonymity model in chapter 2, the distributional privacy model in chapter 3, the graph privacy model in chapter 4, and the personalized privacy model in chapter 5. Related works are discussed in chapter 6, where we also conclude this dissertation, and point out some possible further research directions.

Chapter 2

Anonymizing Numerical data

2.1 Motivation

The privacy goal of k-anonymity[2] and ℓ-diversity[3] 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.

In this chapter, we propose a new privacy goal to better capture privacy protection for numerical sensitive attributes . 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.

It is not hard to see that traditional domain generalization approaches can be used to achieve (k, e)-anonymity when handling numeric-valued sensitive attributes. After each domain generalization, besides checking whether every partition has no less than k tuples, we simply further check whether the partition has a range larger than or equal to e.

However, as we will show in section 2.2, generalization based techniques impose challenges when answering ad hoc aggregate queries. In general, given an aggregate query whose condition over quasi-identifiers is c, and a microdata table generalized under D^*_{QI}, let $\mathcal{D}_i$ be one

Table 2.1: An example microdata table

	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	Carol	40	27130	M	$56,000
4	Debra	41	27229	F	$65,000
5	Evan	43	27269	F	$75,000
6	Frank	47	27243	M	$70,000
7	Gary	52	27656	M	$80,000
8	Henry	53	27686	F	$75,000
9	Ina	58	27635	M	$85,000

subset defined by D^*_{QI}. If for every possible $r = (qi_1, \ldots, qi_k)$ that belongs to $\mathcal{D}_i$ under D^*_{QI}, $c(r)$ is true/false, then obviously all the sensitive attribute values in the partition should be included/excluded when computing the aggregate. Otherwise, to get correct lower/upper bounds of the query result, we have to act conservatively, and include none/all of the sensitive attribute values to compute the aggregate, which can be inaccurate.

In this chapter, we propose permutation-based approaches that can provide the same privacy guarantee, and enable more accurate query answers on the anonymized tables at the same time. Before providing a formal description, we use some examples to illustrate the proposed privacy goal, and the basic idea of permutation-based anonymization and its advantages over a generalization-based approach in the next section.

2.2 Example

Consider the population table shown in table 2.1, which needs to be shared for business and economic research. Among the attributes of the table, "name" can be used to directly identify an individual in real life. To prevent the sensitive information ("salary" in this example) of individuals from being disclosed, such explicit identity attributes need to be removed before the table is disclosed. Other attributes of individuals, such as "age", "zip code" and "gender", often need to be disclosed, since valuable statistical analysis relies on these attributes. For this example, we consider the following aggregate queries:

- **Query 1.** The average salary of those with age over 50.
- **Query 2.** The sum of salaries of those with age between 35 and 55.

Table 2.2: An example 3-anonymity microdata table after generalization.

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

- **Query 3.** The minimum salary of females.

Clearly, after removing the identity attribute "name" from the table in table 2.1, we can still accurately answer these queries.

On the other hand, some public databases may also contain the attributes "age", "zip code" and "gender". They may also further be associated with people's explicit identities. Due to this reason, these attributes are called "quasi-identifiers" since they may be used to reveal one's identity when combined with public databases.

Current approaches to addressing potential privacy violations caused by quasi-identifiers generalize the domains of quasi-identifiers so that many tuples will have the same quasi-identifiers. Table 2.2 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 values of quasi-identifiers. It also satisfies 3-diversity since among those tuples with the same values of quasi-identifiers there are at least 3 different sensitive attribute values.

In this chapter, 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. Table 2.3 shows the table of table 2.1 after permutation, where each group has 3 different sensitive attribute values. In fact, the partition used in table 2.3 is the same as the one resulting from the generalization in table 2.2.

The privacy objective of microdata anonymization is to prevent attackers from knowing sensitive attribute values for an individual. In section 2.3.2, we will show that the permutation-based

Table 2.3: An example 3-anonymous microdata table after permutation

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

approach will achieve the same privacy protection as existing generalization-based approaches. One benefit of our permutation-based approach is that it will provide more accurate answers to aggregate queries.

Let's consider the three queries previously mentioned in this section. Since both approaches introduce imprecise information into the original table, they cannot always get the correct answer for aggregate queries. Instead, for each approach, we can get deterministic lower and upper bounds of the correct answer. We compare the accuracy of the bounds of each approach.

Query 1: The average salary of those with age over 50. This query covers all the tuples in the third group of both table 2.2 and 2.3. Therefore, by using either table, we can get the exact result for the query, which is $80,000.

Query 2: 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 of table 2.2, 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. Therefore, the bounds based on table 2.2 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 of the partition, which aids in the derivation of more accurate bounds. For this query, based on table 2.3, 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 table 2.2.

Query 3: The minimum salary of females. There is a great difficulty in answering this query using table 2.2, since the "gender" attribute is totally suppressed in order to achieve 3-anonymity. We do not even know whether there exists a tuple 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].

Based on the permuted table, we know that there are two female in groups 2 and one in group 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, $70K]. Again, this is more accurate than the bounds derived from table 2.2.

In section 2.3.2, 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.

In fact, since we know the exact number of tuples included in the aggregation in each group (we refer to it as the "number of hits"), more statistics about the query result can be derived. For example, besides the lower and upper bounds for SUM and AVERAGE, we may also compute the mean and variance among all the possible answers. Such statistics provide more information. This is not possible for generalized tables.

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 pre-computes the bounds for each group and all the possible numbers of hits. Given an aggregate query, we can simply rewrite it to query both 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 table 2.1, is shown in table 2.4. Due to space limitations, we only show the bounds for SUM and MIN in the table. It can certainly include those for other aggregation operations.

Both the generalized and the permuted tables satisfy 3-diversity. In fact, they produce the same partition of tuples for anonymization. However, this partition has a problem in terms of privacy. The range of sensitive attribute values in the first group in the partition is only $2,000 while that of others are $10,000. As pointed out in [3], attackers often have external background information about quasi-identifiers of an individual, which enables inferences for the existence of records in the microdata database. In the above examples, 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.

Table 2.4: An example help table

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

Though the attacker cannot know the exact salary of Alex, this range might be narrow enough to be considered as sensitive. This example shows that for numerical sensitive attributes, besides distinct values in each group of a partition, we also need to consider the range of each group to prevent type of inference as the one described. In this chapter we introduce another 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.

We note that several techniques already exist to protect the privacy of numerical sensitive attributes. In perturbation-based approaches [11, 12, 13], noise following a certain distribution is added to sensitive attribute of each tuple. Such an approach inevitably changes important statistics of the marginal distributions of sensitive attributes (e.g., vairance) [14]. Further, depending on the distribution of added noise (e.g., Gaussian Distribution), it is often difficult to derive deterministic bounds for answering aggregate queries.

To some extent, permutation-based anonymization is similar to data swapping techniques where privacy is achieved by exchanging the sensitive attributes of pairs of randomly selected records [15, 16]. Since no noise is introduced, both approaches preserve the marginal distributions of sensitive attributes. However, data swapping is done globally, which has a much larger impact on microdata utility. Even when done in a controlled manner (e.g., rank-based data swapping [17]), it will still produce big errors for aggregate queries. Our experimental results show this point clearly (see section 2.7).

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

2.3 Preliminaries

2.3.1 Privacy in the Release of Microdata

We introduce notations and concepts to facilitate our discussion on potential privacy vulnerabilities in the release of microdata.

Microdata. There are three types of attributes in an *original* microdata table $\mathcal{M}$: identifiers, quasi-identifiers and sensitive attributes. An identifier ID is an attribute, whose value, if known, can *always* be used to *uniquely* identify an individual in real life. In practice, there may exist multiple identifiers such as one's SSN and telephone number. Quasi-identifiers $\{QI_1, \ldots QI_k\}$ are a set of attributes associated with tuples that not only appear in the microdata table but may also appear in other publicly available databases. Example quasi-identifiers include age, date of birth, zip code, etc. Sensitive attributes, on the other hand, are only contained in the microdata table, and do not appear in public databases. The goal of privacy protection is thus to prevent attackers from knowing the specific values of sensitive attributes associated with individual tuples. Without loss of generality, we assume there is only one sensitive attribute S in a microdata table. We further assume that the domain of the sensitive attribute is numeric, which is widely true in a variety of microdata databases. Examples include salary in population databases, credit score in financial databases, and white blood cell count and other diagnosis indicators in public health databases. Table 2.1 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 $\mathcal{S}$.

Note that in some situations the mere fact that there exists a record for a specific individual Alice in the microdata table may also be considered sensitive, even though Alice's sensitive attribute is unknown. For example, the fact that Alice has a record in a microdata table released by a psychiatric hospital may seem quite sensitive. However, as stated in [3], besides public databases, attackers may often have external background knowledge. For example, Bob may physically see that Alice checked into a hospital. Thus, it will be very hard to prevent such information leakage. In my dissertation, revealing one's sensitive attribute values is considered a privacy violation, but revealing the existence of a record with specific quasi-identifiers is not.

Public information. Attackers may often gain access to publicly available information related to individuals. 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-valued, a variety of SQL aggregation operations, such as COUNT, SUM, AVERAGE, MIN and MAX, can be used in aggregate queries.

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, which is based on the one proposed in Yao et al. in [10].

Definition 2.3.1. *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, considering the microdata table in table 2.1, $\{(Alex, \$54,000)$, $(Alex, \$55,000), (Alex, \$56,000)\}$ is a 3-association cover.

Definition 2.3.2. *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 2.3.2, 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 2.2, 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 for numeric-valued sensitive attributes:

Definition 2.3.3. *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.*

2.3.2 Anonymization Through Generalization

Most existing works achieve k-anonymity through domain generalization of quasi-identifiers. That is, instead of releasing the exact values of quasi-identifiers, the values are generalized in a way that many tuples appear to have the same values for quasi-identifiers. For example, instead of disclosing one's exact age, the microdata only shows that the age falls into a certain pre-defined range. Thus, an individual identifier can only be associated with those who have the same quasi-identifiers after generalization.

More specifically, following the notation introduced in [3], let D be a domain. A *generalized domain* D^* of D is a domain $\{D_1, \ldots, D_r\}$, such that each D_i is a subset of D, $D_i \cap D_j = \emptyset$ when $i \neq j$, and $\bigcup D_i = D$. Let $x \in D$. The generalization of x under D^*, denoted $g(x, D^*)$, is $D_i \in D^*$, where $x \in D_i$.

Let $D^*_{QI_i}$ be a generalized domain for each quasi-identifier QI_i. Then $D^*_{QI} = D^*_{QI_1} \times \cdots \times D^*_{QI_k}$ forms a generalized domain for quasi-identifiers. Given a tuple $t \in \mathcal{D}$, its generalization under D^*_{QI}, denoted $g(t, D^*_{QI})$, is a tuple t' such that $t'[QI_i] = g(t[QI_i], D^*_{QI_i})$ and $t'[S] = t[S]$. The generalization of $\mathcal{D}$, denoted $g(\mathcal{D}, D^*_{GI})$, is thus $\{g(t, D^*_{QI}) \mid t \in \mathcal{D}\}$.

In the example shown in table 2.2, the generalization domains for the three quasi-identifiers are $\{[1-10], [11-20], \ldots\}$, the first 3 digits of zip codes, and $\{*\}$ (denoting any gender), respectively.

Given a microdata table $\mathcal{D}$, a generalized domain D_{QI} in fact defines a partition $\{\mathcal{D}_1, \ldots, \mathcal{D}_r\}$ of $\mathcal{D}$ such that any two tuples t_1 and t_2 in $\mathcal{D}$, belong to the same $\mathcal{D}_i$ if and only if they have the same generalized quasi-identifiers under D_{QI}. Let $r = (qi_1, \ldots, qi_k)$ be a value of quasi-identifiers. We say r belongs to $\mathcal{D}_i$ if the generalization of r under D^*_{QI} is the same as that of tuples in $\mathcal{D}_i$.

Most existing works on k-anonymity only require that, after generalization, each $\mathcal{D}_i$ has no less than k tuples. However, as pointed out in [3], this requirement may not always satisfy k-anonymity when the attacker has background knowledge regarding the appearance of individual records in a microdata table. To fix the problem, in [3], it is further required that there should be

at least ℓ distinct sensitive attribute values in each $\mathcal{D}_i$ after generalization. When ℓ is set to be k, the partition obtained by using ℓ-diversity techniques will then guarantee k-anonymity. It is not hard to see that domain generalization can also be used to achieve (k, e)-anonymity when handling numeric-valued sensitive attributes. After each domain generalization, besides checking whether every partition has no less than k tuples, we simply further check whether the partition has a range larger than or equal to e.

However, as shown in section 2.2, generalization based techniques impose challenges when answering ad hoc aggregate queries. In general, given an aggregate query whose condition over quasi-identifiers is c, and a microdata table generalized under D^*_{QI}, let $\mathcal{D}_i$ be one subset defined by D^*_{QI}. If for every possible $r = (qi_1, \ldots, qi_k)$ that belongs to $\mathcal{D}_i$ under D^*_{QI}, $c(r)$ is true/false, then obviously all the sensitive attribute values in the partition should be included/excluded when computing the aggregate. Otherwise, to get correct lower/upper bounds of the query result, we have to act conservatively, and include none/all of the sensitive attribute values to compute the aggregate, which can be inaccurate.

2.4 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 $\mathcal{D}$. Figure 2.1 shows the association between identities and sensitive attributes through quasi-identifiers. 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 our framework, we propose to *only* break 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 2.4.1. *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,*

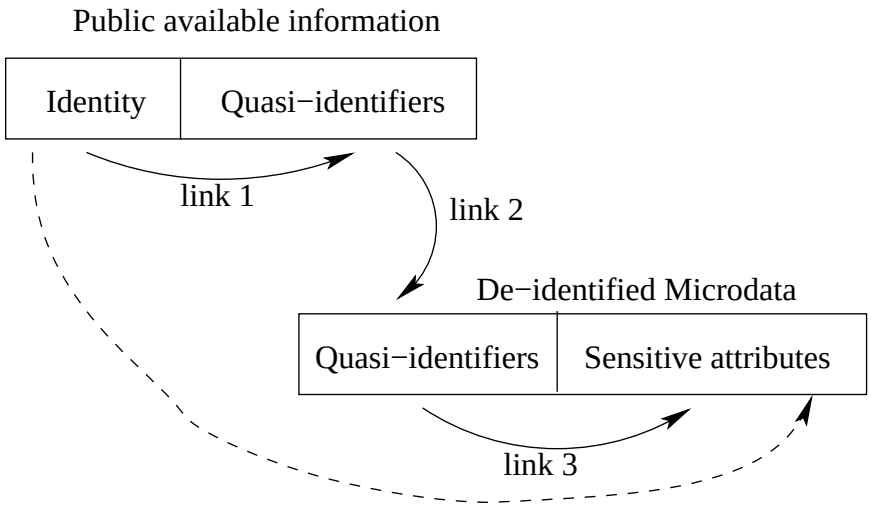


Figure 2.1: The association between identities and sensitive attributes through quasi-identifiers

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 2.4.2. *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 no less than* k *different values, and the range of these different values in* $\mathcal{D}_i$ *is no less than* 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, table 2.3 shows a (3,2000)-anonymous permutation of the table in table 2.1.

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.*

Proof. Prove by contradiction that, if attacker derives an association cover $\mathcal{A}$ of size less than k or with range less than e, then we can construct a microdata table $\mathcal{M}$ such that its (k, e)-anonymous permutation is the same as $\mathcal{D}_p$, and the sensitive attribute of that identifier is not in the association cover. □

In [18], Wong et al. proposed the minimality attack which is effective against almost all the existing generalization-based anonymization schemes. However, we identify that it is not ef-

fective against the permutation-based scheme. The essential reason is that, in order to maximize the utility of the anonymized data, the anonymization scheme often needs to do optimizations on the groupings of quasi identifiers, to create quasi-identifier equivalent classes (QI-EC). The optimizations will leak information about the sensitive attribute distribution in each QI-EC, and can be further used to deduce the actual sensitive value for each tuple, by exploring the generalization criteria during the quasi identifier generalization process. For our permutation-based techniques, we do not group tuples based on the quasi-identifiers, and we do not have such notion of QI-EC. Thus our approach does not have this vulnerability which most generalization-based approaches have. The minimality attack will not work in permutation-based schemes.

2.5 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 when the size of the group is large. Instead, we are interested in efficiently computing important statistics, such as the lower and upper bounds, mean, variance, of all the possible aggregates. Such statistics will be very useful for ad hoc analyses.

2.5.1 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 $\mathcal{D}_i$. Standard SQL aggregation functions, such as SUM, AVERAGE, MIN, MAX and COUNT, are all monotonic. Therefore, the lower and upper bounds of the result of an aggregate query in each $\mathcal{D}_i$ can be efficiently computed.

The lower and upper bounds of the query over the whole microdata table can be derived from those of each $\mathcal{D}_i$, depending on different aggregation functions. For sum, the overall lower/upper bound is the summation of the lower/upper bound of each $\mathcal{D}_i$, respectively. Since we know the total number of tuples satisfying the query condition, the lower/upper bounds of the aver-

age function can be directly computed from those of the sum function. It is also easy to see that the overall lower/upper bounds of the MIN function is the minimum among the lower/upper bounds of all the $\mathcal{D}_i$. The bounds for the MAX function 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 that given by the permuted table using the same partition.*

For sum and average, other statistics besides lower and upper bounds can also be computed by combining those of each group in a partition. We omit details for reasons of space.

2.5.2 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 indicates 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 aggregation operation on the sensitive attribute. Table 2.4 shows the help table for the permuted table in table 2.3. It contains the bounds for SUM and MIN, though in practice it should also contain the bounds of AVERAGE, MAX and other aggregation operations supported by SQL. The number of tuples in the help table is the same as in the microdata table.

Besides the help table, we also create a binary *mapping table* that indicates which tuples the groups of a partition contain.

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 selects the tuple IDs of tuples that satisfy condition C from the permuted table PT. The result in the first step is then joined with the mapping table MT to count the hits of each group of the partition. Once this information is available, it is further 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.

More specifically, for the SUM aggregation operation, we rewrite the query as follows.

1. select groupID, count(tupleID) AS hits into R1
   from MT
   where MT.tupleID in (select tupleID from PT where C)
   group by groupID

2. select sum(sum-lb), sum(sum-ub)
   from HT, R1
   where HT.groupID = R1.groupID and HT.hits = R1.hits

The processing of other aggregate queries is similar.

We emphasize that the help table can be constructed directly from the permuted table (using the mapping 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 and the mapping table once for a released microdata database. It can be done offline, which will not affect the performance of ad hoc analyses.

2.6 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 is unlikely to get satisfactory answers to aggregate queries. Thus, in this section, we turn our discussion to the problem of generating "good" (k, e)-anonymous partitions which are likely 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 2.6.1 (Optimal Partition)**.** *Given a total order D of a sensitive attribute, obtain a (k, e)-anonymous partition $P = \{G_1, \cdots, G_m\}$ that minimizes $F(E(G_1), E(G_2),$*
$\cdots, E(G_m))$ 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 is 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, it seems imperative 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.

2.6.1 The Minimum Sum-of-Error Problem

Recall that the goal of this problem is to find a (k, e)-anonymous partition $P = \{G_1, \ldots, G_m\}$ of D such that $sum\_of\_error(P)$ is minimized. 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 2.6.1. *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 $max_i < min_{i+1}$.*

Proof. We first observe that $max_i < max_{i+1}$. Otherwise, since $min_i < min_{i+1}$, we can simply merge G_{i+1} and G_i into one group, whose maximum error is still $x_{max_i} - x_{min_i}$, and obtain another (k, e)-anonymous partition P'. We have $sum\_of\_error(P') = sum\_of\_range(P) - (x_{max_{i+1}} - x_{min_{i+1}})$, contradicting the fact that P is optimal.

Second, if $max_i > min_{i+1}$, we can still merge G_i and G_{i+1}. The maximum error of the new group is $x_{max_{i+1}} - x_{min_i}$, which is less than or equal to the sum of the maximum errors of G_i and G_{i+1}. □

Lemma 2.6.1 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

$$f(i) = min_{1 \leq d \leq i} F(f(d-1), E(\{x_d, \ldots, x_i\})) \tag{2.1}$$

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) the optimal m^*-1 partitioning of $x_1, \ldots x_{d-1}$ (for some d, $1 \leq d \leq i$) with the group $\{x_d, \ldots, x_i\}$. Algorithm 1 presents the dynamic programming solution for the sum-of-error problem. The algorithm considers all values of $i, 1 \leq i \leq n$, and for each value of $d, 1 \leq d \leq i$, assesses the sum of errors using equation 2.1. The index of the minimum item of each group is stored in the array *partition*. A linear scan of this array at the end of the algorithm (starting from partition[n] and going backwards) will extract the optimal group descriptions. The algorithm makes use of the array $distinct$ that returns in $O(1)$ the number of distinct elements in D between the arguments supplied. The size of this array is $O(n^2)$ and can be populated in a preprocessing step in $O(n^2)$ time, so that access to it remains $O(1)$. It is evident that Algorithm 1 runs in $O(n^2)$.

Algorithm 1 Optimal partition for the minimum sum-of-error problem

```
f[0] = infinity
partition[0] = 0
for i = 1 to n do
    f[i] = infinity
    partition[i] = partition[i-1]
    for d = 1 to i do
        if distinct({x_d, ..., x_i}) ≥ k and x_i − x_d ≥ e then
            error = x_i − x_d
        else
            error = infinity
        end if
        temp = max(f[d-1], error)
        if temp < f[i] then
            f[i] = temp
            partition[i] = d
        end if
    end for
end for
```

Theorem 3. *Given a total order of a sensitive attribute D of n items, with $O(n^2)$ preprocessing and $O(n^2)$ space, we can compute in $O(n^2)$ time, the optimal partition of D for the sum-of-error*

problem.

Proof. The proof is straightforward following the above analysis. □

2.6.2 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 $x_{max_i} = x_{min_{i+1}}$). Therefore, each group can be described completely by using the indices of its first and last attribute values min_i and max_i, and every attribute value with index between them belongs to the group. 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 2.6.2. *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.*

Proof. First, by a similar proof to that of lemma 2.6.1, no group's range is included by that of another.

Second, given any optimal partition, suppose there are three groups G_i, G_j and G_l, $i < j < l$, whose ranges overlap with each other. Then we must have $x_{max_j} \geq x_{min_l}$. Otherwise, since $x_{max_i} \leq x_{max_j}$, we have the ranges of G_i and G_l are not overlapping. We thus can divide all the items in G_j into two groups, G_{j1} including those less than x_{min_l}, and G_{j2} including those greater than or equal to x_{min_l}. We merge G_{j1} with G_i and G_{j2} with G_l. The ranges of G_i and G_l do not change, which means the new partition is still optimal. By repeating this step, we will get an optimal partition where the ranges of no three groups overlap with each other. □

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 2.6.2, 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, ..., 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 $E(G'_1) + r$, $E(G'_2)$, ..., $E(G'_z)$ 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 2.6.2. *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 $g(d, r, i) = min_{i \le u \le n, i \le v \le u-1} max(g(d - \#(i, u, v, d), r - (x_{u+1} - x_v), u + 1), x_u - x_i + r)$

Algorithm 2 Optimal partition for the minimum max-of-error problem

```
for i = n to 1 do
  for d = 0 to distinct(1, n) do
    for j = 0 to i do
      r = x_i − x_j
      if distinct({x_i, ..., x_n}) < k − d or x_n − x_i < e − r then
        g[i][d][r]=INF
        continue
      end if
      g[i][d][r]=x_n − x_i
      for max_1 = i to n − 1 do
        if distinct({x_i, ..., x_{max_1}) < k − d then
          continue;
        end if
        for min_2 = i + 1 to max_1 + 1 do
          d_1 = #(i, max_1, min_2, d)
          r_1 = x_{max_1+1} − x_{min_2}
          m = max(x_{max_1} − x_i + r, g[max_1 + 1][d_1][r_1])
          if m < g[i][d][r] then
            g[i][d][r] = m
          end if
        end for
      end for
    end for
  end for
end for
```

Algorithm 2 shows the dynamic programming solution to the minimum max-of-error problem. For simplicity, algorithm 2 only returns the maximum error of an optimal partition. With some simple bookkeeping, the algorithm can be easily modified to return the items contained in each group.

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 very

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. In other words, we consider the following problem:

Problem 2.6.3. *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 complexities.

Theorem 4. *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 $min\_max\_error(P')$ $\leq 2 \cdot min\_max\_error(P)$.*

Proof. Let $P = \{G_1, \ldots, G_m\}$. According to lemma 2.6.2, overlapping ranges can only happen between adjacent groups. We examine each group by order. If the range of G_1 does not overlap with that of G_2, we continue to G_2. Otherwise, G_2 can be divided into two groups, G_{2f} which includes those no less than x_{min_3}, and G_{2l} which includes those greater than x_{min_3} (G_{2l} may be empty if the ranges of G_2 and G_3 do not overlap). We merge G_{2f} into G_1 and G_{2l} into G_3. Note that the range of G_3 does not change, while the range of G_1 is at most increased by $max\_of\_error(P)$. We next move to G_3 and check whether it overlaps with G_4. We continue this process until G_m. The resulting partition P' does not contain overlapping groups, and $max\_of\_error(P')$ is at most twice $max\_of\_error(P)$. □

2.7 Experiments

Our experiments are conducted on the Adult Database from the UCI Machine Learning Repository [19]. 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, 3]. We choose the same quasi-identifiers (which contain 8 attributes) in our experiments 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, so that we can adjust a variety of parameters to comprehensively evaluate the properties of the proposed permutation-based approach. The synthetic data set uses the same schema as the Adult Database. We populate the database with different numbers of tuples, assuming certain distribution of the capital loss attributes. We also consider the correlation between the capital loss attribute and quasi-identifiers. The details of the synthetic data set will be described later when we present the experimental results.

We design three sets of experiments to study the following aspects of different k-anonymity techniques.

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.

Query answering overhead. As described in section 2.5, to answer an aggregate query Q over a permuted database, we first rewrite it into a query Q' which queries the permuted table, the mapping table and the help table. Our second set of experiments measure the running time of Q', and compare it with the time that it takes to execute Q over the un-permuted microdata table. The difference shows how much overhead our technique introduces to answer aggregate queries.

Scalability of partitioning algorithms. We have shown in section 2.6 that the optimal algorithm for the minimum sum-of-error problem and the approximation algorithm for the minimum max-of-error problem are of complexity $O(n^2)$. In this set of experiments, we empirically show their scalability when increasing the size of the microdata table.

Next, we describe each set of experiments in detail.

2.7.1 Accuracy

We first compare the relative error 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 [3]. 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. This significantly limits the type of queries the generalized table can answer. To make 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, in order to prevent "age" from being generalized to "*" when using the ℓ-diversity algorithm, we also have to remove some of the outliers (6 tuples with age over 80). Otherwise, the "age" attribute will still be suppressed even with 4-diversity.

The resulting 4-diverse partition is composed of 25 groups. 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*. Arbitrary queries are representative of various disjunctive queries. From a generalized table, we cannot know for sure how many tuples are actually selected by a query in each group. To get deterministic lower and upper bounds, we have to assume either no tuples are included in the query, or only the tuple with the maximum capital-loss is selected. On the other hand, from a permuted table, we can always know exactly the number of tuples in each group that are selected by the query. Figure 2.2 shows the relative errors of the bounds derived from the generalized table and the permuted table. We see that the relative errors introduced by the generalized table is significantly higher than that by the permuted table. In fact, the bounds from the generalized table are often over two times of the actual query results. 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.

We recognize that, since the arbitrary query is a very general model, the generalized table will not be able to take advantage of the semantics of a query condition. 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 $[X, Y]$. Clearly, for the generalized table, if the generalized value of the age attribute of a group falls completely in $[X, Y]$, then all the tuples in the group should be selected for the

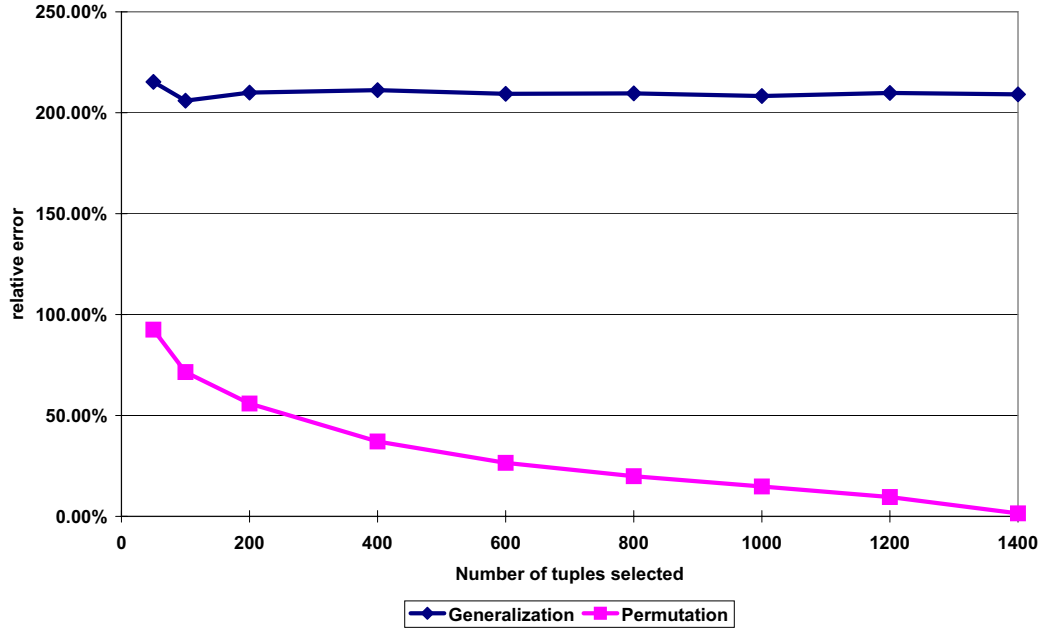


Figure 2.2: The relative errors of arbitrary queries when using the generalized table and the permuted table respectively

aggregation. Otherwise, we have to derive the lower and upper bounds of the group as in the first experiment. For the permuted table, we will still know the exact number of tuples selected in each group. The relative errors for range queries are shown in figure 2.3. We see that the accuracy of the bounds derived from both the generalized table and the permuted table improve for range queries. In particular, the relative error computed from the permuted table is less than 20% even when the span of the range query is only 5, which does not completely cover any groups in the partition. For the generalized table, the error is still very high when the range of the query is small, because some groups are only partially covered by the query. Since we do not know exactly how many tuples are selected by the range query in a partially covered group, the lower and upper bounds will be quite coarse. 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 less attributes, the hierarchy of generalization 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 2.2 and 2.3

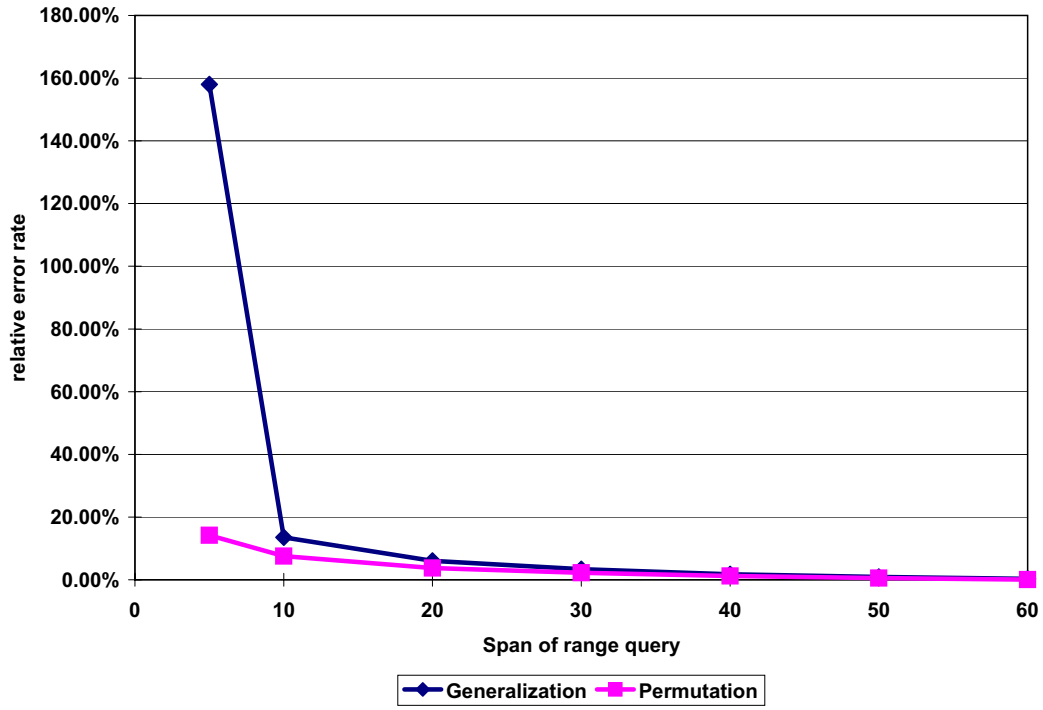


Figure 2.3: The relative errors of range queries when using the generalized table and the permuted table

(see figure 2.4 and 2.5). Our observation suggests that the poor query answering accuracy from the generalized table is often not caused by high dimension of quasi-identifiers. Instead, it is intrinsic to the domain generalization approach.

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 2.6): (1) ℓ-diversity based on domain generalization; (2) Min Max: the approximation algorithm for the minimum max-of-error problem; (3) Min Sum: the optimal algorithm for the minimum sum-of-error problem; (4) Max Group: an algorithm that generates the maximum number of groups in the partition. Similar to the minimum sum-of-error proble, the optimal algorithm can be obtained through dynamic programming; (5) a random algorithm: this algorithm sequentially scans each tuple. 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; and (6) the rank-based data swapping algorithm [17]. To be comparable to (k, e) anonymity, given a tuple t whose rank is $rank(t)$, we select l such that the set of tuples with ranks in $[rank(t) - l, rank(t) + l]$

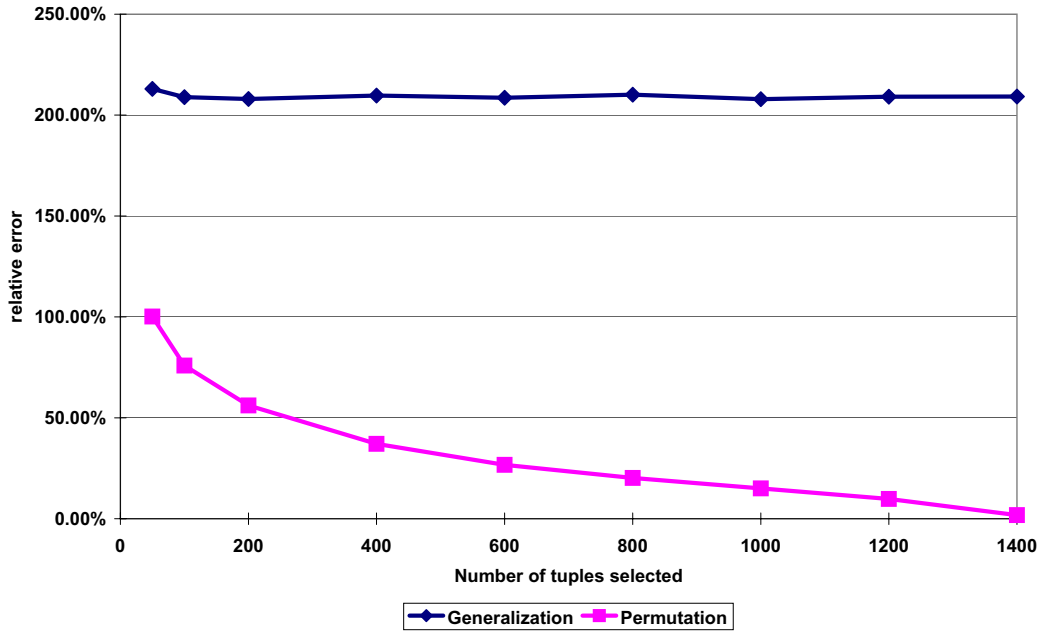


Figure 2.4: The relative errors of range queries when using the generalized table and the permuted table with only one QI

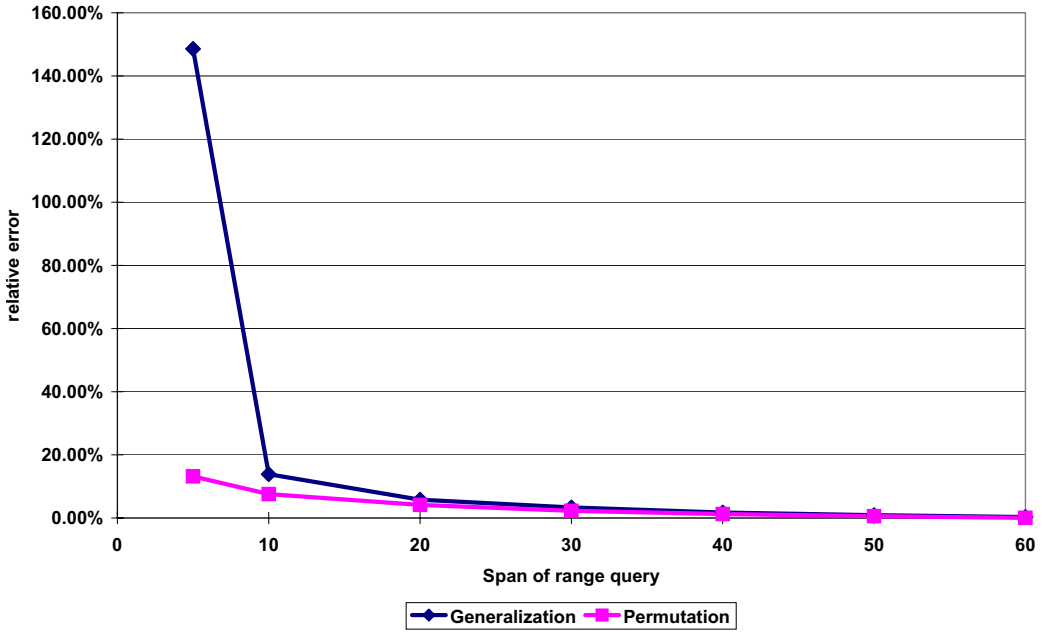


Figure 2.5: The relative errors of range queries when using the generalized table and the permuted table, with only one QI

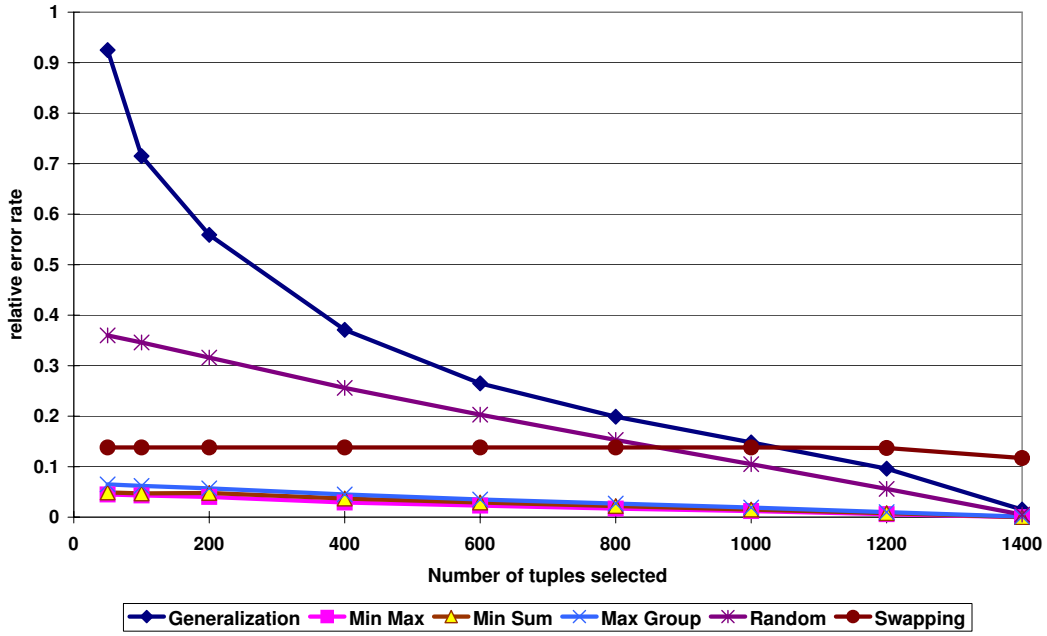


Figure 2.6: The relative errors of arbitrary queries when using different partitioning algorithms

satisfies (k, e) anonymity. We then swap t with a random tuple whose rank is in the range.

We set $k = 4$ as before, and set $e = 100$ for the two optimization algorithms and the random partitions. To be in favor of 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 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 2.6 (for arbitrary queries) and figure 2.7 (for range queries). It is clear that the three optimization algorithms introduce significantly less relative errors than the other two algorithms for arbitrary queries. This can be easily explained since the two optimization algorithms are not constraint by pre-defined domain hierarchies. They have more flexibility to partition the table and achieve better accuracy. We also observe that the partition from generalization is even worse than the random partition for arbitrary queries. This shows that the partition derived from pre-defined generalization hierarchies greatly reduces the utility of microdata, even if we use permutation-based anonymization. The rank-based data swapping algorithm, though with better accuracy than that from the partitions obtained through generalization hierarchy, still introduces much large relative errors than the the three optimization algorithm.

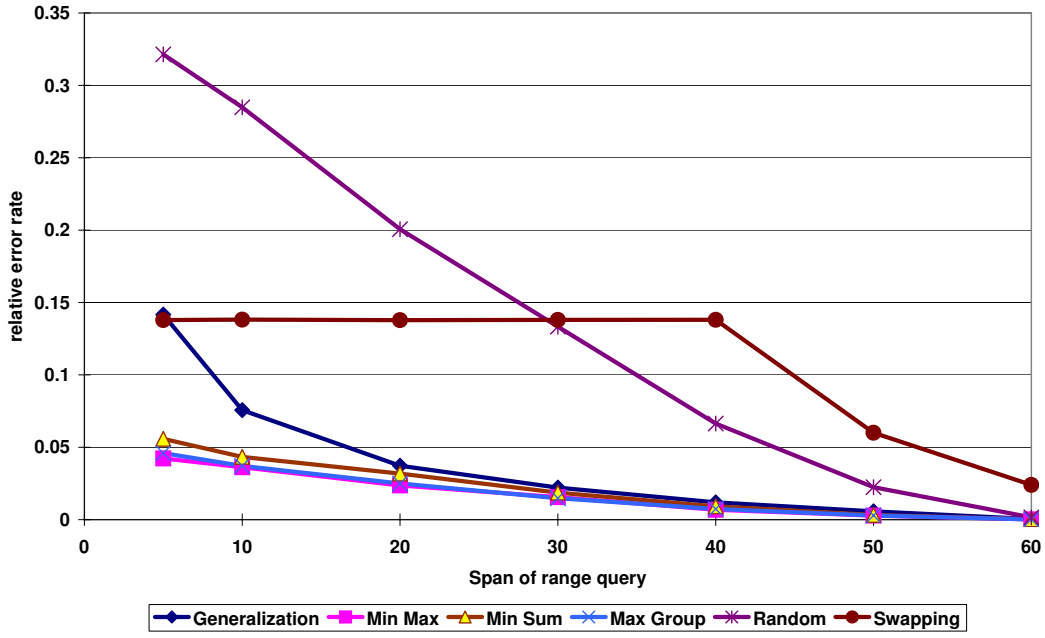


Figure 2.7: The relative errors of range queries using partitions from different partitioning algorithms

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 2.8 shows the relative errors of the six 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 domain 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

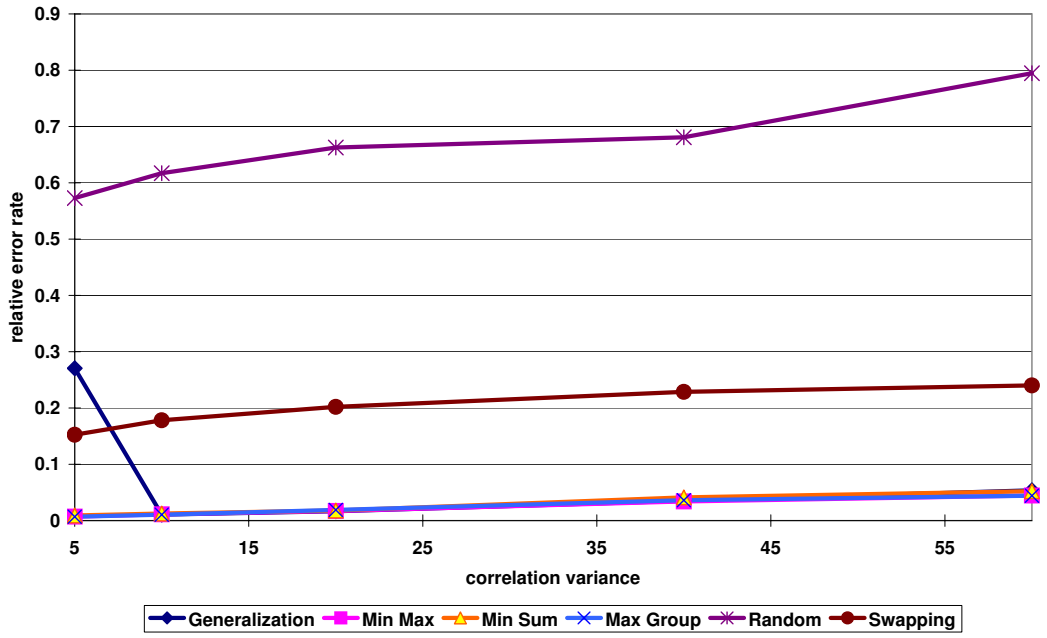


Figure 2.8: The relative errors of range queries using partitions from different partitioning algorithms, when the strength of the correlation between quasi-identifiers and the sensitive attribute varies.

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. The rank-based data swapping algorithm does not perform very well because it is hard to deterministically reason the bounds of a query after data swapping. Only when a long sequence of consecutive sensitive attribute values are covered by a query, can we say for sure that some tuples are definitely included in the original query answers. This only happens frequently when a large portion of tuples are touched by a query.

Finally, we study 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 introduces more errors when answering aggregate queries. To see the tradeoff more clearly, we run experiments over a synthetic random data set, whose quasi-identifier and sensitive attribute are "age" and "capital-loss" attributes respectively. The ranges of them are the same as in the adult database. We issue range queries over "age" attribute. The span of range queries is set to 30 as before. To examine the impact of the privacy parameter k, we fix $e = 50$, and vary k from 4

to 40. We measure the relative errors of the partitions obtained by the two optimization algorithms and the random partitioning for each k [1]. Figure 2.9 shows the experiment's results. We see that the accuracy for the partitions obtained by the three optimization algorithms are essentially the same, and 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. Meanwhile, we see that the accuracy of the range-based data swapping approach decreases significantly, since it does not provide a mechanism to minimize the error introduced by data swapping.

We next evaluate the impact of the other privacy parameter e. In this experiment, k is fixed to be 10, and e varies from 40 to 200. Figure 2.10 shows the same trend as in figure 2.9: a much larger privacy requirement e does not impact the accuracy of the partitions obtained by the two optimization algorithms. As for the random algorithm, the increase of e does not affect the generated partition much, since random distinct values in the same group usually result in a range much larger than the privacy parameter e set by the experiment. The range-based data swapping approach does not perform well due to the same reason as explained in the previous experiment.

2.7.2 Query Execution Overhead

Given an aggregate query Q, we compare its execution time over the original un-permuted microdata table with that of the rewritten query Q' which performs selection and joins over the permuted table, the mapping table and the help table. To make the comparison more clearly, we run the experiments on the synthetic data set with 10000 tuples. The partition is $(20, 0)$-anonymous, and is obtained by the first optimization algorithm. There are totally 38 groups in the partition. Both arbitrary queries and range queries are tested.

In order to study the impact of the number of tuples and partitions involved in a query on query execution overhead, we use constrained arbitrary queries in this experiment: we vary the number of partitions involved in the query while having the percentage of tuples selected in each group fixed to be 30%. In figure 2.11, we show the time of the two steps when executing a rewritten query. The first step is to query the permuted table and join it with the mapping table so that the number of tuples selected in each partition is obtained. In the second step, the result from the first step is joined with the help table and the lower and upper bounds are computed. We also show the

[1] Partitions from generalization is not studied because the original ℓ-diversity does not have the parameter e. Moreover, we observe that when we require 6-diversity, in the real database "age" is already generalized to "*", and it cannot tell any information about a query on "age".

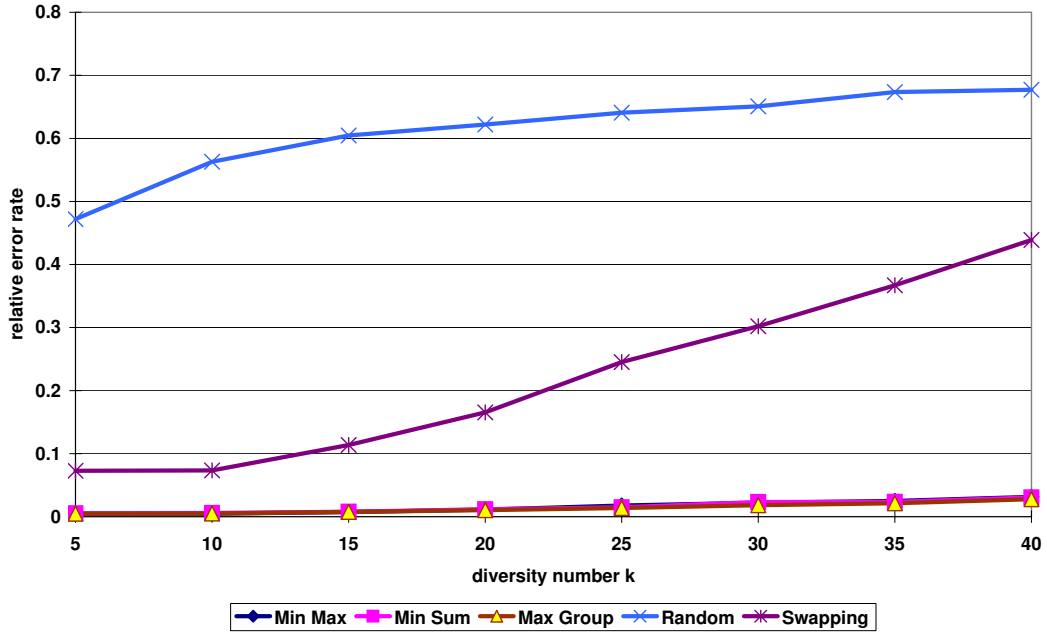


Figure 2.9: The relative errors of range queries using partitions from different partitioning algorithms

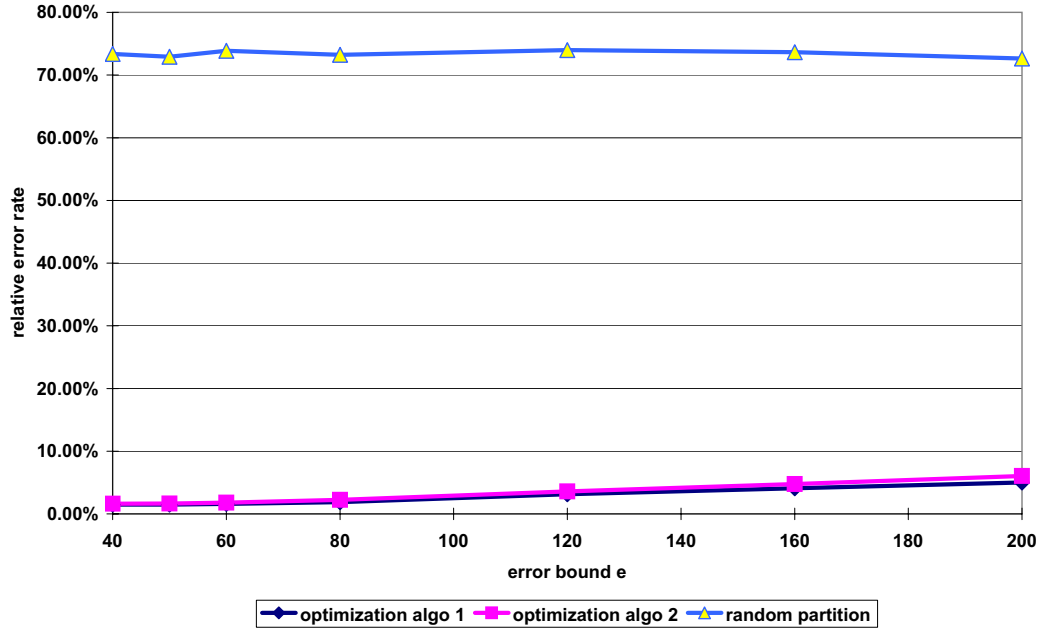


Figure 2.10: The relative errors of range queries using partitions from different partitioning algorithms

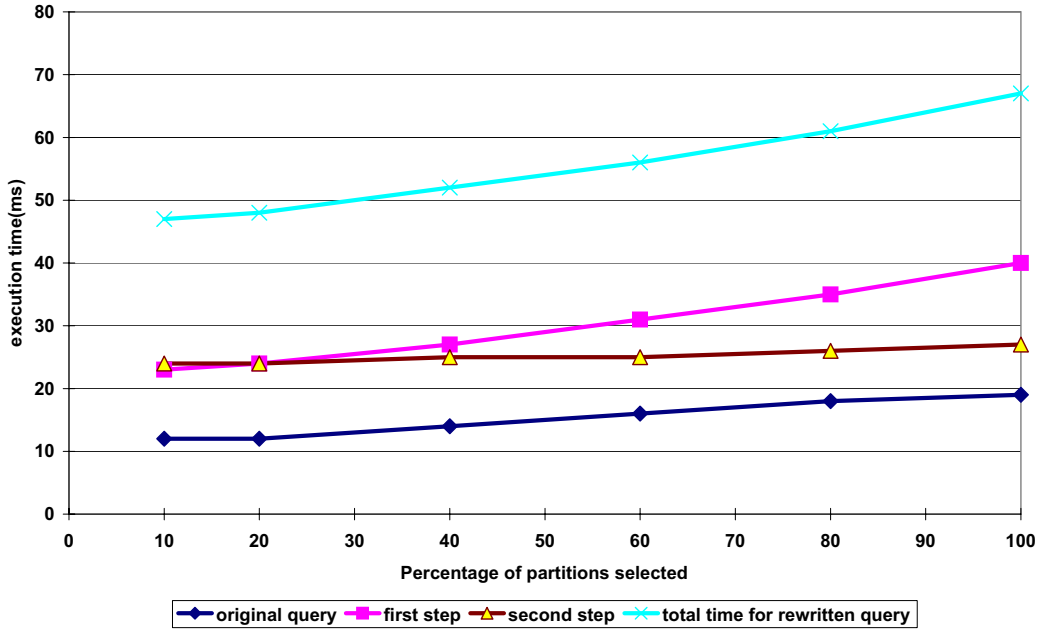


Figure 2.11: The execution time of the constrained query and the rewritten controlled query when the percentage of selected groups in the partition varies

total running time of the rewritten query and compare it with the case if we run the original query directly on the un-anonymized microdata table. We see that, when the number of involved groups increases, so does the running time of both steps. For the first step, it is because the number of tuples selected in the first steps grows. For the second step, the more groups involved, the longer it takes to finish the join with the help table. The overall execution time of the rewritten query is about four times that of the original query.

The range query takes the same form as in the previous experiments. We increase the span of its range from 10 to 50. Intuitively, the larger the span of the range, the more groups of the partition and tuples will be selected by the query. Figure 2.12 presents the experiment's result, which is consistent with the experiment using constrained queries.

2.7.3 Scalability

We study the scalability of the optimization algorithms presented in section 2.6. All of them have computational complexity of $O(n^2)$, and have similar performance. Figure 2.13 shows the running time of the first partitioning optimization algorithm while the database size is varied from 100 tuples to 2000 tuples.

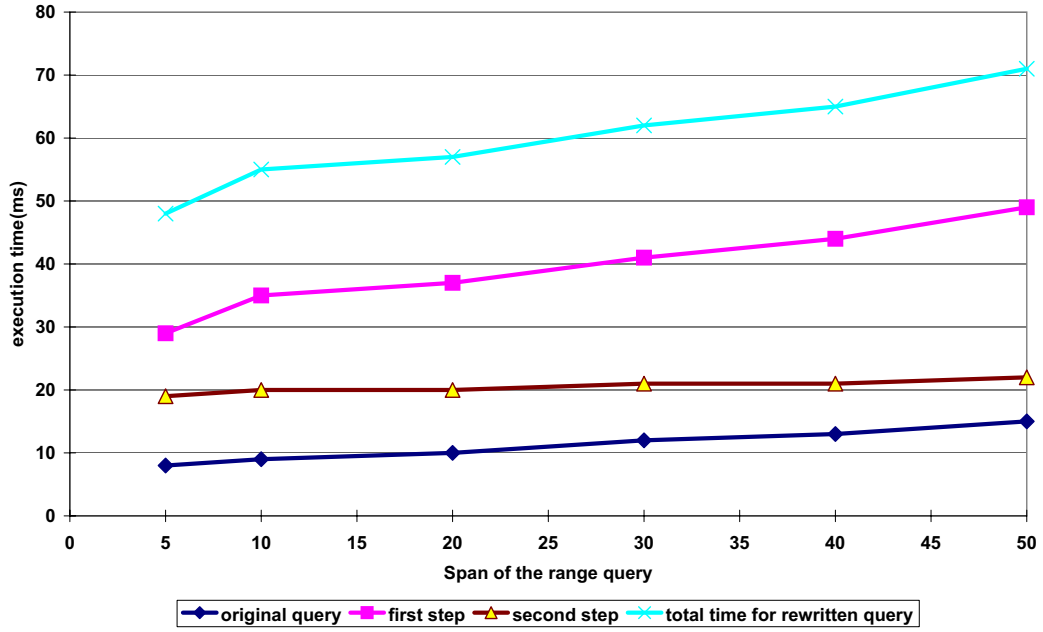


Figure 2.12: The execution time of the original range query and the rewritten range query when the span of the range varies

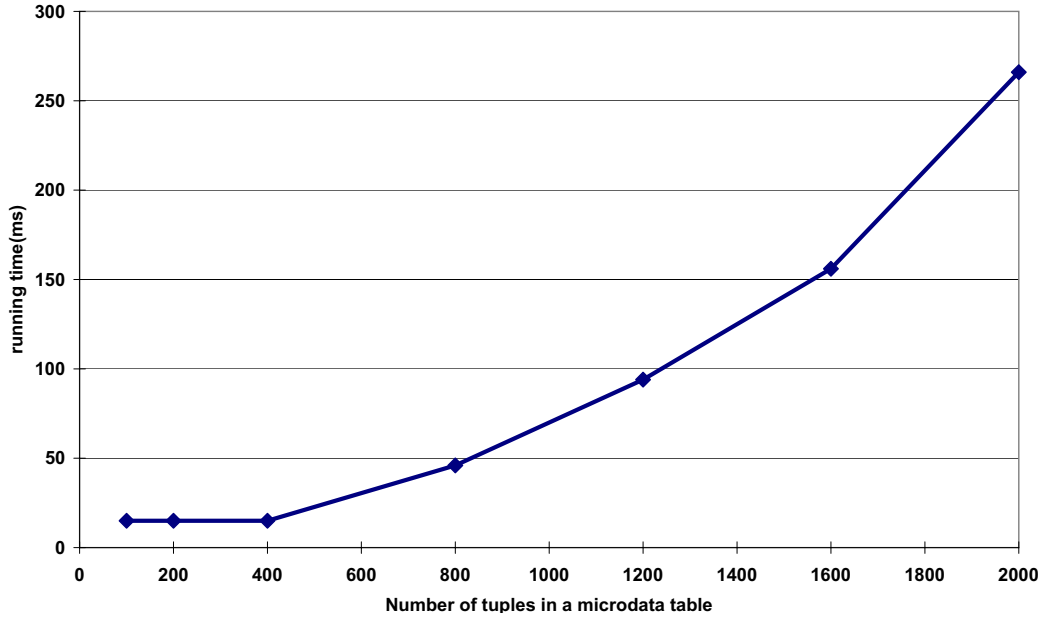


Figure 2.13: The running time of an optimal partitioning algorithm when the microdata table size scales up

2.8 Summary

In this chapter, we propose a permutation-based anonymization schemes to perform microdata protection. We show that microdata anonymized via the permutation-based based schemes will have much better utility than traditional generalization-based schemes. And it is effective against the minimality attack.

We further propose an extended privacy objective to better capture the protection of numeric-valued attributes in microdata. We show that we can achieve the same privacy guarantee as existing work when we partition a microdata table and perform random permutation between quasi-identifiers and sensitive attributes inside the groups of the partition. Further, since the quasi-identifiers of tuples remains in the anonymized table, aggregate queries can be answered much more accurately. We also design auxiliary relations and query rewriting algorithms to facilitate efficient ad hoc analyses over anonymized tables.

Chapter 3

Disguising Data Distribution

3.1 Motivation

In the previous chapter, we propose an ad hoc privacy model which provides privacy guarantee for numerical data. As we have seen it successfully increases the protection for numeric data. However, in some situations, attackers may know some background information about the sensitive attributes. For example, attackers can find out the distribution of the salary for all IT professionals in the U.S aged between $[30, 35]$ from the internet. Suppose it is a normal distribution with mean $70K$. After publishing the (k, e)-anonymity table, the salary value in a certain group falls in the range $[90K, 100K]$, and there is a person whose age is in the range of $[30, 35]$. Then for this person, his privacy is compromised in some sense.

Generally, we observe that in many scenarios the *distribution* of specific attribute values in the released microdata can be as sensitive as the private information of individual records. For example, consider a hospital in Beverly Hills that treats celebrities. Since reporters often keep monitoring which celebrities checked into the hospital, it would be desirable for the hospital to disguise the distribution of the released sensitive attributes (e.g., symptoms and prescriptions) to a different "target distribution" which, e.g., exactly or approximately mirrors the distribution from the general population of patients in the Los Angeles area.

Sometimes, the distributional information of only a part of the microdata may be sensitive. As an example, a company may secretly adopt an aggressive policy to attract and keep employees with certain skills (e.g., security), which causes the salary distribution of those employees to be significantly different from that of other similar companies. Thus, the company may want to disguise the salary distribution of a particular portion of its employee records so that outsiders cannot

infer its current strategic policies. For instance, when sharing its employee records, the company may choose to transform this portion of employee salaries to follow a target distribution similar to a distribution considered typical in the industry.

From the above examples, we see that there are two desirable properties when protecting sensitive distribution information of microdata. First, it is often the case that not all the distributional information is sensitive. Hence, a protection scheme should allow the microdata owner to indicate exactly the parts of the data whose distribution should be disguised. Second, the microdata owner should have the flexibility to disguise different portions of the data to exactly or approximately match different target distributions. For example, while the distribution of the general population of patients in the Los Angeles area is an appropriate target distribution for a hospital in Beverly Hills, it is not so for hospitals in Washington DC.

Unfortunately, existing approaches to microdata protection cannot achieve the above goals. The exact distribution information of microdata will always be revealed in both quasi-identifier generalization [7, 3, 20] and sensitive attribute permutation [21, 22], two major techniques in microdata protection. Further, there are no mechanisms available to selectively disguise the distributions of particular parts of the released data to match an arbitrary target distribution.

Besides disguising the actual distribution, it is also important to maintain as much useful information as possible in the microdata. In particular, it should still be possible to answer ad hoc aggregate queries with reasonable accuracy. Otherwise, the whole purpose of data sharing becomes meaningless. For instance, one simple approach to disguising a distribution is to replace each record's sensitive attribute value with one that is randomly generated from the target distribution. However, this approach changes the original data completely, and does not have any deterministic guarantee on the accuracy of query answering.

In this chapter, we propose a new privacy model that provides supports for flexible privacy requirements. In other words, we allow the owner of microdata to flexibly constrain what information an attacker can derive about the sensitive attribute of an individual. Though existing works such as l-diversity and t-closeness, or our previous (k, e)-anonymity model do provide parameters to adjust privacy requirements, they do not provide sufficient flexibility to the microdata owner. As an example, if the microdata owner would like an attacker to infer nothing more about an individual's sensitive attribute value than the fact that it could be any value in the domain of the attribute with equal probability, irrespective of the actual distribution of sensitive attribute values in the original microdata table, none of the existing privacy definitions can meet this need.

In order to achieve this privacy goal, we propose a variation of the basic SA-permutation

Table 3.1: An example microdata table

	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

scheme. We first generalize the sensitive attributes according to a predefined hierarchy, then we can group them and permute the generalized domains within each hierarchy. As previous the design goal is still to give accurate deterministic bounds of the actual query result when the query is executed over the original microdata which is not anonymized.

Next, we use some examples to illustrate the proposed privacy goal, and the basic idea of the new anonymization scheme and its advantages over a generalization-based approach.

3.2 Example

Table 3.1 shows an example employee record table of a company. Before this table is shared for business analysis, the explicit identity attribute Name should be removed. Attributes Zipcode and Gender are quasi-identifiers. If an individual can be uniquely identified when an attacker combines its quasi-identifiers with publicly available information, then the attacker can discover his/her sensitive attribute Salary. For simplicity, we assume that the domain of Salary is $\{30K, 40K, 50K, 60K\}$.

There are two concerns when releasing the above microdata. First, outsiders should not discover the exact salary of an individual, even with the help of some external knowledge. For example, an attacker may know that his neighbor Nancy is working for the company. Thus, even if we remove the name attribute and only keep the first three digits of the zipcode attribute, he can still figure out that Nancy's salary is likely to be 60K (as most people with zipcode 913** are with salary 60K).

Table 3.2: An example microdata table after sensitive attribute generalization and permutation

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

Second, we can also observe that employees from those areas with zipcode 913** are paid significantly higher than those from other nearby areas. This trend may be related to the company's sensitive strategic policies of resource allocation. Thus the company may have the motivation to disguise the salary distribution so that it appears to be, for example, a uniform distribution from {30K, 40K, 50K, 60K}, which is typical for employees working for other companies in that area. This goal cannot be achieved by existing anonymization techniques. Even though recent works in microdata protection take the distribution of sensitive attributes into consideration [3, 7], they cannot transform the sensitive attribute values to follow an arbitrary given target distribution.

Our basic scheme is to leave quasi-identifiers unchanged and generalize and permute the sensitive attribute values. One unique feature of this scheme is to give data owners the flexibility to determine the distributions of which parts of the data should be protected and how. The microdata owner may partition the microdata into multiple groups, and specify the *target distribution* that each group's sensitive attribute values should appear to follow. This allows a much finer control of information sharing. For example, if the distribution of one part of the microdata is not considered sensitive, the target distribution may just be its original distribution. For some other part, the target distribution may be a typical one among the industry, among a geographical area or worldwide. Our scheme does not impose any restrictions on target distributions.

Table 3.2 shows one possible anonymized microdata table after applying our scheme. In this example, we partition the table according to the first three digits of the zipcode attribute. For simplicity, we assume that the target distributions for all the groups are the uniform distribution over {30K, 40K, 50K, 50K}. Figure 3.1 shows the generalization hierarchy. As we will show later,

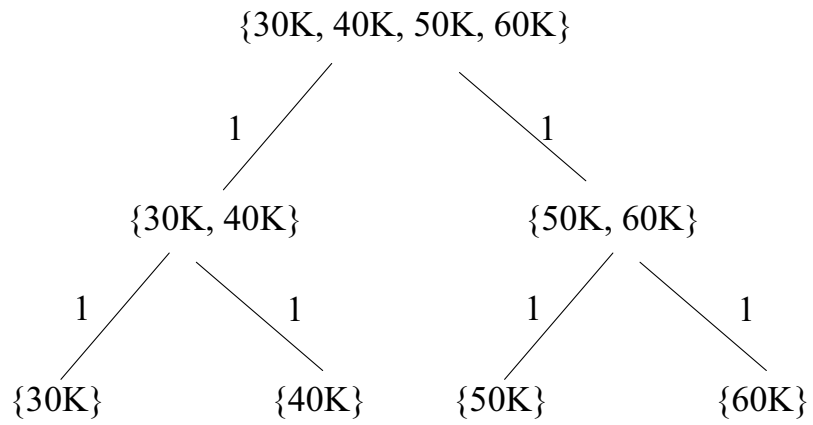


Figure 3.1: An example domain generalization hierarchy

any target distribution can in fact be specified over the domain generalization hierarchy by giving different weights between the children of the nodes in the hierarchy tree. The weight assignments in figure 3.1 actually correspond to a uniform distribution.

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 Table 3.2, each tuple's sensitive attribute is generalized to itself. These values are then permuted among group 1. In group 2, the two sensitive attribute values 40K are generalized to {30K, 40K}, while the other two sensitive attribute values are unchanged. Then again we perform a 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 but nothing more.

Clearly, 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 queries:

Query 1: What is the average salary of female employees?

Query 2: What is the average salary of employees living in the area with zipcode 91110?

To answer Query 1, 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, for the generalized value {30K, 40K}, as 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 bounds for the tuple from group 3 would be [30K, 60K]. Overall, the bounds for the answer of query 1 would be [35K, 55K]. Similarly, for Query 2, the bounds would be [40K, 50K].

Meanwhile, because of sensitive attribute generalization and permutation, we can easily verify that the distribution of each tuple's sensitive attribute follows the uniform distribution, which means that privacy of individuals are effectively protected. As a contrast, approaches based on quasi-identifier generalization cannot provide such a strong protection.

We note that, given a group of sensitive attributes, there are many ways to generalize them so that after permutation the target distribution is followed. For example, table 3.3 shows another way of generalization and permutation to achieve the same target distribution for each group. Another extreme example is to generalize the sensitive attribute of each tuple to the whole domain. Though they all offer the same protection of sensitive information, the accuracy of query answering from all these different schemes may vary greatly. In fact, it is not hard to verify that the bounds offered derived from table 3.3 and from the trivial generalization table are much worse than that from table 3.2.

Intuitively, the less generalization we perform, the better bounds we tend to obtain. Meanwhile, the more clustered quasi-identifiers of tuples in a group are, it seems the better, especially for range queries. Based on the above observation, in this paper, we identify optimization metrics and design algorithms which produce sensitive attribute generalization and permutation schemes that offer better query answering accuracy.

Another important technique we propose to improve the accuracy of query answering while protecting distributional information 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 table 3.2, which contains records of those employees in area with zipcode 912**. With only generalization and permutation, it is easy to see that the generalization shown in table 3.2 is the best we can do in terms of minimizing the sum of ranges of generalization. If we ask what is the average salary of employees in that area, the best bound we can get is $[110K/3, 160K/3]$.

Table 3.3: An example microdata table after sensitive attribute generalization and permutation, where aggregate queries cannot be accurately answered.

		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}

Table 3.4: Inserting fake values into microdata

		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}

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 one. Thus we only need to perform a permutation without any generalization. Group 2 will then be transformed into table 3.4.

Note that 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]. It can be easily verified that from the above published table one can only infer that the possible salary of each employee is uniformly distributed between $30K$ and $60K$, but nothing more. 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 $[40K, 50K]$.

We observe that adding fake values may reduce the level of generalizations needed to match a target distribution. On the other hand, fake values also introduce uncertainty into the origi-

[1] In practice, 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. This technique has been used in previous works [21].

nal data set. If we do not have any restriction on the number of fake values, given any data set, we can always transform it to match a target distribution without using any generalization. However, this may not be the best choice in terms of the accuracy of aggregate query answering. For example, in group 3 of Table 3.2, 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 $[130K/4, 250K/4]$, which is worse than $[140K, 220K/4]$, the bound if we only apply generalization and permutation without adding any fake values. Thus, it would be sensible to minimize the sum of ranges of generalizations while limiting the number of added fake values. In this paper, we will present an efficient algorithm to do so.

3.3 Disguising Data Distribution

3.3.1 Basic Concepts and Notations

Without loss of generality, we assume the schema of a de-identified microdata table is composed of two parts: quasi-identifiers $\{Q_1, \ldots, Q_k\}$, and sensitive attributes S. In our work, we focus on only one sensitive attribute. When there are many, we can treat each attribute independently using our techniques on one attribute. The problem becomes tricker when the multiple sensitive attributes are correlated. We note that the previous related work [21, 22] has no good solutions for this case either, and we leave it for future work. We focus on numerical data since they support more interesting aggregation operations, such as SUM and AVERAGE, which are not applicable to categorical data.[2]

Given the domain $\mathcal{D}(S)$ of S, we say $\mathcal{D}' \subset \mathcal{D}(S)$ is a subdomain of S if it covers a continuous segment of $\mathcal{D}(S)$. 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) $\mathcal{D}(\mathcal{S})$ is the root node; (3) all the children of a node N form a partition of N; and (4) a leaf node is a subdomain $\{t\}$, where $t \in \mathcal{D}(S)$. Without loss of generality we assume H is a binary tree.

As mentioned in section 3.2, 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 is to make sure that in the released microdata table, one can only observe that the sensitive attribute value of each group follows its target distribution. Note that we do not require the target distribution to be

[2] It is not hard to see that our techniques can be easily adapted to handle categorical data.

the same for all groups. 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 disguise the distribution of each group independently.

From an anonymized table T, if we can only derive that the distribution of the sensitive attribute of each tuple follows a distribution P, then we say T is *P-private*. For example, the first and third groups of the microdata table in table 3.2 are U-private, where U is the uniform distribution over {30K, 40K, 50K, 60K}. To prevent the attacker from inferring sensitive information from the target distribution, the target distribution P is generally modeled as the best knowledge an attacker may have about a sensitive attribute, given all the external knowledge. For example, the attacker may already know the distribution P' of the salaries of software engineers in the United States. From the anonymized table, the attacker may be able to have a better guess of the salary of a particular software engineer in the company. This situation generally is not desirable. The microdata owner may thus specify P' to be the target distribution, which ensures that an attacker cannot gain any additional information.

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. In the rest of this chapter, unless otherwise noted, we assume that a target distribution is given as a privacy-annotated hierarchy.

3.3.2 Generalization and Permutation

As shown in section 3.2, the basic idea of our approach is to generalize the sensitive attribute value of each tuple to a subdomain of $\mathcal{D}(S)$. We then randomly permute the resulting subdomains among all the tuples, so that the distribution of the possible values 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 3.1, 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 (i.e., the hierarchy corresponds to a different target distribution). Then if $t.S \in$ {30K, 40K}, then the probabilities of $t.S$ to be 30K and 40K will be 2/3 and 1/3

respectively.

Definition 3.3.1. *Let $G = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a multiset of subdomains of $\mathcal{D}(S)$, and P be a target distribution over $\mathcal{D}(S)$. We say G is P-*private *if for any element $t \in \mathcal{D}(S)$, $P(x = t) = \frac{1}{n}\sum_{1 \le i \le n} P(x = t | x \in \mathcal{D}_i)$.*

For example, both {{30K}, {40K}, {50K}, {60K}} and {{30K, 40K}, {30K, 40K}, {50K}, {60K}} are U-private.

Definition 3.3.2. *Let $g = \{s_1, \ldots, s_n\}$ be a multiset of elements in $\mathcal{D}(S)$, $G = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a multiset of subdomains of $\mathcal{D}(S)$, and P be a target distribution over $\mathcal{D}(S)$. We say G is a P-*private generalization *of g, if G is P-private and there is a one-to-one mapping f from g to G, such that $s_i \in f(s_i)$, $i = 1, \ldots, n$.*

For example, both {{30K}, {40K}, {50K}, {60K}} and {{30K, 40K}, {30K, 40K}, {50K}, {60K}} are U-private generalizations of {30K, 40K, 50K, 60K} while only {{30K, 40K}, {30K, 40K}, {50K}, {60K}} is also a U-private generalization of {30K, 30K, 50K, 60K}.

Given a group of sensitive attributes, there are many different P-private generalizations. A trivial P-private generalization is to generalize each sensitive attribute value to the whole domain $\mathcal{D}(S)$. Though secure, the resulting microdata table is hardly useful. It is thus important to develop algorithms to produce generalizations that protect privacy, and also answer aggregate queries accurately. Before presenting such algorithms, we next discuss how to answer aggregate queries over an anonymized microdata table.

3.3.3 Answering Aggregate Queries

As what we have described in the (k, e) anonymity model, we focus on aggregate queries that select a subset of tuples based on arbitrary conditions on quasi-identifiers and then aggregate over the sensitive attributes, i.e., they are of the form "select Aggr(S) from Microdata where C", where C is a condition on quasi-identifiers, and Aggr may be common aggregations such as SUM, AVERAGE, MIN, MAX. Such queries are commonly used in ad hoc data analyses. The detailed query answering scheme has been proposed in section 2.5, and we will not further elaborate on the details here. But it is worth pointing out that we have to do a minor extension to allow for sensitive attribute values that are generalized to subdomains. This is not needed in techniques that rely only on permutation.

3.4 Accurate Query Answering Optimization

In this section we present optimization criteria to disguise distributional information of microdata so that aggregate queries can be more accurately answered.

3.4.1 Minimize the Sum of Ranges of Sensitive Attribute Generalizations

The higher a subdomain is in the domain hierarchy, the more "uncertainty" is introduced when answering queries. A natural measure of such uncertainty is its range, i.e., the difference between the maximum and the minimum elements in the subdomain[3]. Formally, we define the following optimization problem.

Problem 3.4.1. *Given a multiset g of sensitive attribute values, a privacy-annotated domain generalization hierarchy H that corresponds to a distribution P over $\mathcal{D}(S)$, 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.

To facilitate our discussion, we introduce the following notation. 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 $g_{\mathcal{D}'} = \{30K, 30K, 40K\}$.

Definition 3.4.1 (m-bound allocation)**.** *Let $N = (n_1, \cdots, n_k)$ be a sequence of integers, $W = (w_1, \ldots, w_k)$ be a sequence of weights, and m be an integer. An m-bound allocation of N subject to W is a set of integers $(n'_1, \ldots, n'_k)$, where $n'_i \leq n_i$ for $i = 1, \ldots, k$, and $n'_1 : \cdots : n'_k = w_1 : \cdots : w_k$, and $\sum_{1 \leq i \leq k} n'_i \leq m$.*

We say $(n'_1, \ldots, n'_k)$ is a *maximum m-bound allocation*, if no other m-bound allocation has a larger sum. 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 $N = (3, 6, 5)$, $W = (1, 2, 1)$ and $m = 10$. Then $(2, 4, 2)$ is the maximum m-bound allocation of N subject to W. If $m = 3$, then the only m-bound allocation of N subject to W is $(0, 0, 0)$.

To minimize the sum of ranges of generalization, it is preferable to generalize a sensitive attribute value to a smaller subdomain in the privacy-annotated hierarchy. Meanwhile, to be P-private, the number of sensitive values generalized to a certain level of subdomains should preserve

[3]For categorical data, the uncertainty is determined by the number of elements in a subdomain. We thus can treat its cardinality as its range.

the ratio between the weights among them in the privacy-annotated hierarchy. This explains the intuition of applying maximum allocation to solve problem 3.4.1.

Algorithm 3 shows an optimal solution to this problem. Initially we set $m = |g|$. 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 can 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.

Algorithm 3 MinSumOfRange(g, H, m): optimal generalization for the minimum sum of range problem

$\mathcal{D}$ = the root of H
$\mathcal{D}_1, \ldots, \mathcal{D}_k$ = the children of the root
$W = (w_1, \cdots, w_k)$ = weights of $\mathcal{D}_1, \ldots, \mathcal{D}_k$ in H
$N = (|g_{\mathcal{D}_1}|, \cdots, |g_{\mathcal{D}_k}|)$
$(n_1, \ldots, n_k)$ = maximum m-bound allocation(N,W)
Output $m - \sum_{i=1,\ldots,k} n_i$ $\mathcal{D}$ to the final generalization
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

For example, consider the privacy-annotated hierarchy in figure 3.1 and a multiset g of sensitive attribute values {30K, 30K, 40K, 40K, 50K, 60K}. 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, 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. This means that any P-private generalization should contain at least two root domains. After we have decided how many subdomains under each subtree should be contained in the final generalization, we recursively continue the same process for each subtree.

Observe that as we continue to the children of a node by calling the function Min-

SumOfRange, we pass the subtree rooted at a child node, and the new bound m. Meanwhile, we still pass g, all the tuples in the group, to the function. In other words, in our algorithm, it is not necessary 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 domains we have to have in an optimal solution. This is the essential reason for the efficiency of the algorithm.

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 has 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. This makes it practical for very large microdata tables.

Theorem 5. *The multiset of subdomains generated by 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.*

Proof. At each step of the algorithm 3, we push any many values towards the leaf nodes of the tree as possible. So the algorithm processes at each step such that the values generalized to the root of each subtree is minimized. Suppose there's an another algorithm A^* that is optimal, and produced a generalized G^* which is different from the output of G_1 produced by Algorithm 3. As we require that the generalization should follow the hierarchy, that is, the generalized domains must be from the nodes of the hierarchy tree. We can then count the number of generalized domains of each node in G^*, and then compare it from the root with that of G_1. Suppose the first difference is at some node N_i in height i. It must be that G^* has α more values generalized to the domain of N_i than G_1. Now if we keep the other parts of G^* unchanged, and simply push the α values to the subtrees following the process of Algorithm 3, the sum of ranges will be reduced. This suggests that G^* is not optimal. Thus we can prove that Algorithm 3 produces the optimal solution. The complexity analysis is already shown as above. □

3.4.2 Minimize Range of 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 Table 3.2. 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. These observations suggest 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 instead take a pragmatic approach. Specifically, we first partition a group into multiple subgroups such that tuples in each subgroup have the same quasi-identifiers. And then for each subgroup we find its optimal P-private generalization. We call this algorithm the *QI-SA* algorithm. As a contrast, we use *SA-only* to denote the approach that only uses the optimal sensitive attribute generalization to answer queries without further partitioning.

One advantage of this approach 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, even 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 answering than the other. In our experiments, we will perform a detailed comparison of the two approaches with different data sets.

3.5 Integrating Fake Values into The Framework

In this section, we consider the use of fake values to further improve query answering accuracy. As shown in section 3.2, adding fake values may reduce the level of generalization needed to match a target distribution. 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 be beneficial to get accurate 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 optimization

problem.

Problem 3.5.1. *Given a multiset g of sensitive attribute values, a privacy-annotated hierarchy H corresponding to a 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.*

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 $\mathcal{D}$.

Though in general the more fake values we add, the less generalization we need to match the target distribution, this trend is not monotonic. For example, 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). 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 $minSOR(g \cup f_2, H) > minSOR(g \cup f, H)$.

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 $g \cup f$ 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 $g \cup f_1$ 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 $g \cup f_2$ 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 3.5.1 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 complication of the problem, let us consider a simpler case. Suppose we only allow to add a single fake value to g, and the target distribution is uniform. Assume in g the number of values in one subtree (assuming left subtree) is one more than that in the other subtree (assuming 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, then we

can solve the problem 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. For example, consider g=$\{1, 1, 1, 2, 3, 4, 5, 6, 7, 9, 9, 11, 11, 13, 13, 15, 15\}$, where the domain $\mathcal{D} = \{1 \dots 16\}$. 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.

Next we present an optimal algorithm for problem 3.5.1. 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 3.5.1 is in fact 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 *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 the subproblem $minSORFake(g_{\mathcal{D}_2}, x, r_2, P_{\mathcal{D}_2})$, where $r_2 = |g_{\mathcal{D}_1}| + (k - x) - v$.

In summary, we have

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

where u, v, r_1 and r_2 are calculated as described above. Algorithm 4 shows the pseudocode for the optimal algorithm. For simplicity, we present the algorithm in the form of recursion, which can be easily converted to a standard dynamic programming algorithm.

Algorithm 4 minSORFake(g, k, r, H): the minimum sum of ranges of generalization 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

To solve problem 3.5.1 completely, we have to 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. Let $g = \{1, 2, 3, 5, 6, 7, 7\}$ and $D = \{1, 2, \ldots, 8\}$. Assume H is a balanced binary tree that represents the uniform distribution over 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 $[5 - 6]$, 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($g_{[1-4]}$, $H_{[1-4]}$, $|g_{[1-4]}|$).

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.

Complexity Analysis. Suppose the hierarchy H is a balanced binary tree, and the domain of the sensitive attribute is D (i.e., the root domain of H). Observe that there are at most $2|D|$ 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 $2(t+1)(|g|+t)|D|$. For each cell, the computation takes the minimum over $t + 1$ possible cells, which gives the time bound as $O(t^2(|g| + t)|D|)$.

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 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 D', a tighter bound is given by $min((t+1)^d, |g_{D'}| + t)$.

Summed over all the 2^d nodes in the same level as 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 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 $|g|(1+log|D|)+ 2t|D|$. 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 $(t+1)(|g|(1+log|D|)+ 2t|D|)$, which is much tighter than the bound $2(t+1)(|g|+t)|D|$ given above. Similarly, a much tighter bound for time complexity is given by $O(t^2(|g|log|D| + 2t|D|))$.

3.6 Approximate Target Distributions

Our discussion so far assumes that an anonymized table has to follow the target distribution exactly. However, in many situations this may not be necessary. It is usually not sensitive for an outsider to know, for example, that a company's salary distribution is not exactly the same as that of the whole industry (there is probably no such company). What is sensitive is the magnitude of the difference. Strictly requiring the distribution P_{anony} after anonymization to be the same as the target distribution P_{target} may significantly reduce the quality of microdata. Hence, it would be desirable to have a mechanism so that a microdata owner can control how much the target distribution can be approximated to tradeoff support for more accurate data analysis.

Recall that we use a privacy-annotated tree to model a target distribution, where the weights of two sibling subdomains show the probabilities for a tuple's sensitive attribute value to fall into the subdomains. Given a privacy-annotated tree, it is thus natural to quantify the difference between P_{anony} and P_{target} in terms of this ratio. In detail, a data owner may specify a *shift factor* $x \in [0,1]$. P_{anony} is acceptable for privacy protection if for any node N in the hierarchical tree, the weights w_1 and w_2 of N's children in P_{target} and the weights w_1' and w_2' in P_{anony} satisfy $|\frac{w_1}{w_1+w_2} - \frac{w_1'}{w_1'+w_2'}| \leq x$ and $|\frac{w_2}{w_1+w_2} - \frac{w_2'}{w_1'+w_2'}| \leq x$.[4]

[4] A straightforward extension is to specify a different x for each node. Our algorithm can be equally applied to this control mechanism.

Given the shift factor x, a set S of sensitive attribute values and P_{target}, we can easily compute P_{anony} results in the least generalizations after anonymization among all acceptable distributions. Essentially, we shift the weight of each node in P_{target} subject to x to make it as close to the actual distribution as possible.

Specifically, given two children subdomains $\mathcal{D}_1$ and $\mathcal{D}_2$, whose weights are w_1 and w_2 in P_{target}, if $\frac{|g_{\mathcal{D}_1}|}{|S|} \leq \frac{w_1}{w_1+w_2}$, then we set w_1' and w_2', the weight of $\mathcal{D}_1$ and $\mathcal{D}_2$ in P_{anony}, such that $\frac{w_1'}{w_1'+w_2'} = \frac{|g_{\mathcal{D}_1}|}{|S|}$ if $\frac{|g_{\mathcal{D}_1}|}{|S|} > \frac{w_1}{w_1+w_2} - x$, or $\frac{w_1'}{w_1'+w_2'} = \frac{w_1}{w_1+w_2} - x$ otherwise. The case when $\frac{|g_{\mathcal{D}_1}|}{|S|} > \frac{w_1}{w_1+w_2}$ is similarly handled.

Once P_{anony} is obtained, we can apply algorithm 3 on P_{anony}. Intuitively, since P_{anony} is the closest to the actual distribution, the anonymization process will introduce the minimum generalization in terms of sum of the ranges of the generalized subdomains.

3.7 Experiments

Our experiments are conducted on the Adult Database from the UCI Machine Learning Repository [19]. The same database has been used in previous works [6, 3], and we choose the same quasi-identifiers (which contain 8 attributes). We choose "capital loss" as the sensitive attribute as we focus on numerical data in this section. 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.

We also conduct experiments on synthetic data sets, so that we can adjust a variety of parameters to comprehensively evaluate the properties of the proposed anonymization scheme. The synthetic data sets use the same schema as the Adult Database. We populate the database with data from different distributions. We also consider the correlation between the sensitive attribute and quasi-identifiers. Details will be described later as appropriate.

We design the following experiments to verify the effectiveness of our approach, and study the impact of the target distribution and the generalization hierarchy on the accuracy of query answering. In all the experiments, we evaluate the accuracy of the bounds derived from a microdata table protected by the proposed sensitive attribute generalization and permutation technique. Specifically, let l and u be the lower and an upper bounds 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.

The performance of anonymization through SA generalization. In this set of exper-

iments, we show the accuracy of aggregate query answers of different query sizes, when applying the SA-only algorithm and QI-SA algorithm on both real and synthetic datasets.

The impact of adding fake values. In this set of experiments, we show the impact of integrating different number of fake values with the QISA and SAonly schemes.

The impact of the target distributions. In this set of experiments, 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. In this set of experiments, we show the effectiveness of our algorithm when we enforce different target distributions for different parts of the data.

Privacy-accuracy trade off. In this set of experiments, we study the tradeoff between privacy and accuracy by studying the resulted sum of ranges of generalization and query accuracy under different shifting parameter x.

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.

3.7.1 Performance of Anonymization through Sensitive Attribute Generalization

We first study the relative errors of our algorithms applied on different datasets. In all experiments in this section, the target distribution is set as the source distribution of sensitive attribute values in the overall microdata table. 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 3.2(a) shows the relative errors of the bounds derived 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 produces quite inaccurate bounds. The reason is that when the query range is small, the selected tuples only count for a

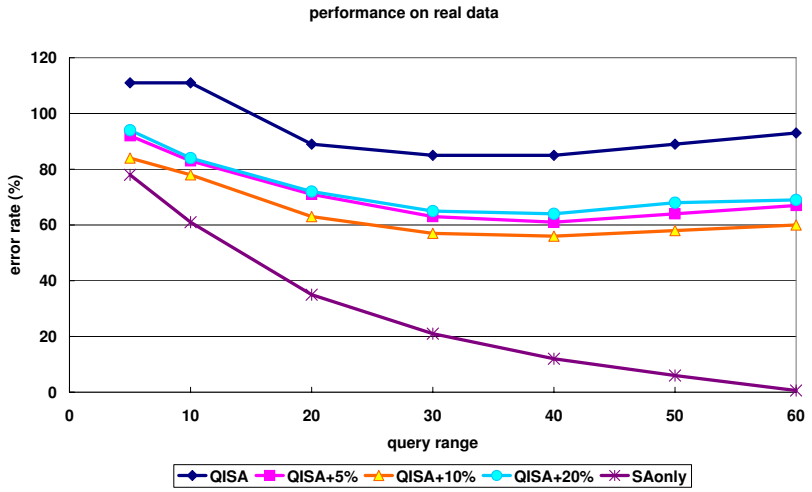
very small portion of each group, which results in bounds with poor accuracy. When the query range increases, as SA-only minimizes the sum of ranges across all generalized subdomains, it produces bounds with better accuracy than QI-SA.

Meanwhile, as QI-SA generates much 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 overall target distribution. So even when the query range increases, the accuracy of the bounds does not improve.

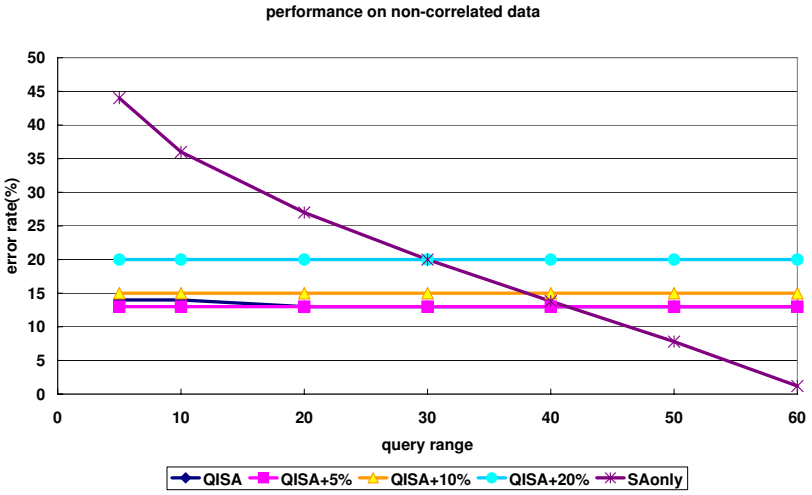
To further validate our evaluation, we compare the accuracy of QI-SA and SA-only on two synthetic data sets, with different sensitive attribute distributions and correlations between quasi-identifiers and the sensitive attribute. In the first dataset, there is no correlation between quasi-identifiers 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 3.2(b).

From the figure we can see that when there is no correlation between quasi-identifiers and the sensitive attribute, 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 sensitive attribute generalizations are needed to achieve the privacy requirement. 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 of the sensitive attribute.

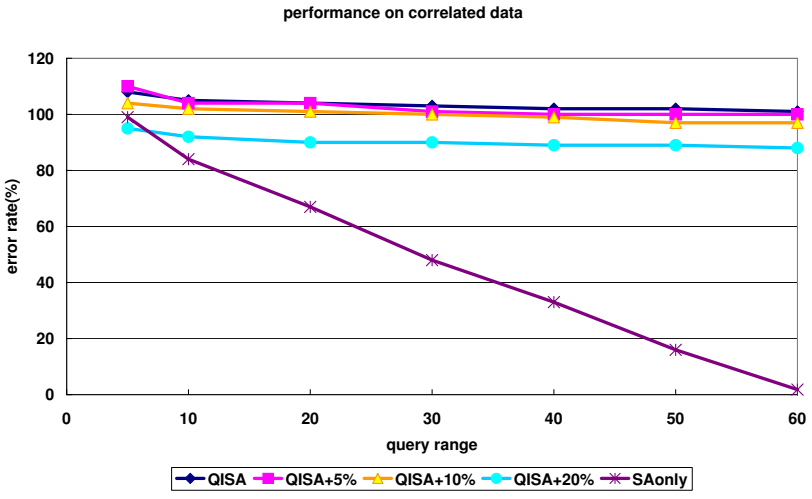
Intuitively, if there is a strong correlation between quasi-identifiers and the sensitive attribute, tuples in each age group generated by the QI-SA algorithm tend to have similar sensitive attribute values. Therefore, they need to be generalized to higher subdomains in the hierarchy in order to follow the target distribution. Our next experiment aims to investigate the impact of correlation on the accuracy of query answering. We compare QI-SA and SA-only when varying the correlation between quasi-identifiers and the sensitive attribute. In the second synthetic dataset, 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". Therefore, 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. In more detail, we divide the domain of "capital loss" equally into 79-17+1=63 buckets. For a tuple with age a, the mean v of the corresponding normal distribution is set to be the central point of bucket $a - 17$. The size of the



(a) Real data



(b) Non-correlated data



(c) Correlated data

Figure 3.2: Relative error rate of QI-SA and SA-only, and QI-SA with fake values

dataset is also 10k. Figure 3.2(c) shows the relative errors of QI-SA and SA-only over this data set.

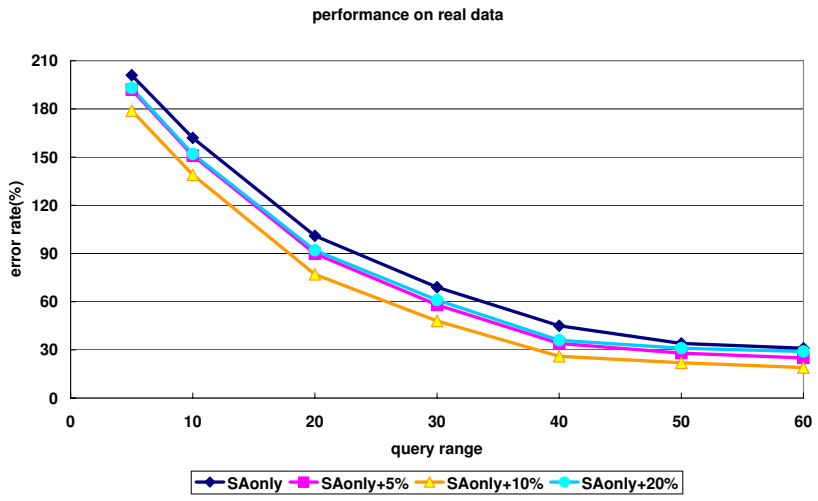
We see that SA-only consistently outperforms QI-SA. In fact, the trend of SA-only is similar to the case when there is no correlation between quasi-identifiers and the sensitive attribute. On the other hand, QI-SA, as expected, performs poorly, since it only considers the sensitive attribute values inside a single group, whose distribution is far 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, when quasi-identifiers and the sensitive attribute are correlated, SA-only tends to answer aggregate queries more accurately. In the case when quasi-identifiers and the sensitive attribute are not correlated, QI-SA offers more accurate query answering for smaller query ranges.

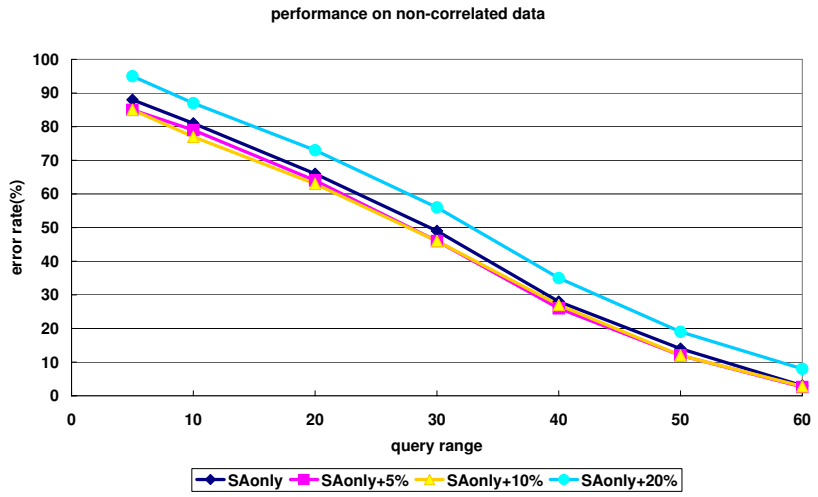
3.7.2 Impact of Integrating Fake Values

In this section we will study the impact of integrating fake values with the QI-SA and SA-only schemes. First we look at the QI-SA scheme. In each QI partition of size N, we allow up to $K = x\% \cdot N$ fake values. Figure 3.2 shows the results on the real data set and the two synthetic data sets. Different number of fake values allowed are denoted in the legend as "$+x\%$". Figure 3.2(a) shows the impact in the real database. When adding a few fake tuples, it can help reduce the error rates effectively. But when adding too many fake tuples (20%), the fact that more SA values than QI values in each group begin to dominate, and query accuracy is getting worse. Figure 3.2(b) shows the impact in the non-correlated synthetic data. Where there's no correlation between QI and SA, in each partition the SA will be very close to the target distribution, so there are very few generalizations initially. When adding few fake tuples (5%), it helps to reduce the error rate. But when adding more fake tuples, more SA values than QI values in each group begin to dominate quickly, and query accuracy is getting worse. Figure 3.2(c) shows the impact on the correlated synthetic data. There are many generalizations initially in this case. When adding more fake tuples, it keeps reducing the error rate. One major reason lies in the target distribution: In this case all ratios in the hierarchy are 1:1, so adding more values can effectively push more nodes to lower levels of the hierarchy tree.

Next we look at the SA-only scheme. When the target distribution is the same as the source distribution, we know that the SA-only scheme is the same as global permutation, and no generalization is involved. So adding fake values will not have effect. Thus we need to modify our simulation setup so that we can study the impact of integrating fake values with the SA-only



(a) Real data



(b) Non-correlated data

(c) Correlated data

Figure 3.3: Relative error rate of SAonly with fake values

scheme. First we will randomly remove 10% of the source data. The source distribution is then different with the target distribution, and some sensitive attributes have to be generalized. We also notice that the ratio of the source distribution may contain some big values initially, which may result in massive generalizations. For example if the ratio is 1000:1001, and the actual counts in the two subtrees are 100:100, then all the 200 values has to be generalized to the root. To avoid such case, we also normalize the ratios in the tree to the ranges of 10. Figure 3.3 shows the results. We have exactly the same observations as those of the QI-SA case.

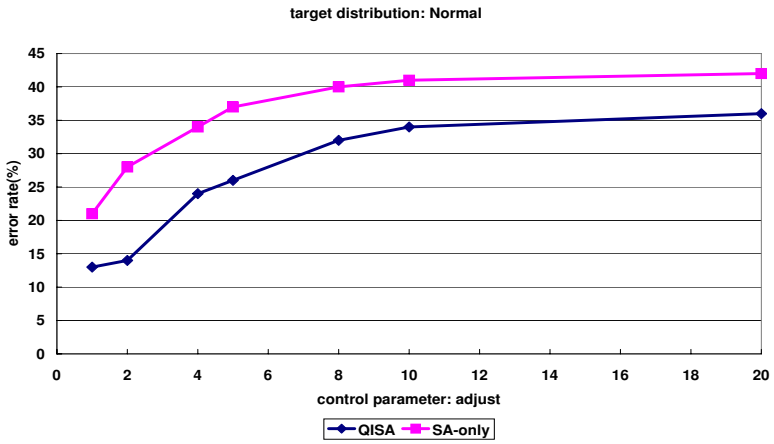
3.7.3 Impact of Target Distributions

In the following experiment, we investigate the relative errors of QISA and SA-only when we have different target distributions and domain generalization hierarchies. As QISA does not perform well when there is a strong correlation between quasi-identifiers and the sensitive attribute, in this experiment we use a non-correlated dataset.

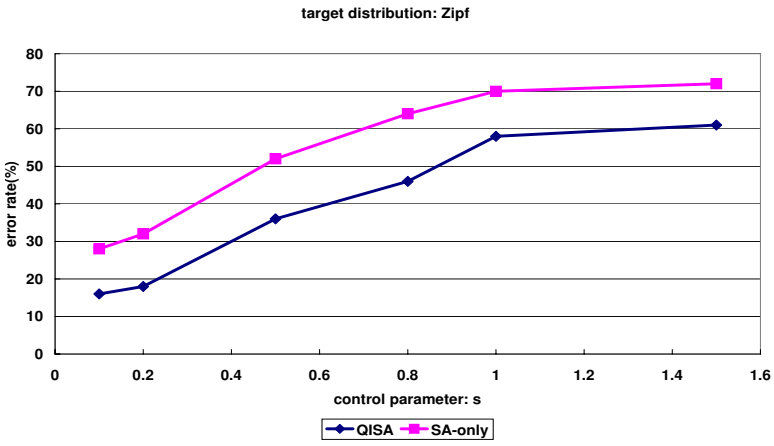
The first experiment studies the trends when the target distribution is different from the actual distribution. In the synthetic dataset of size 10k, the sensitive attribute "capital loss" is generated following a uniform distribution in the range $D = [1024, 3072]$. We then set the target distribution to be a normal distribution. Its variance equals $range(D)/r$, where r is a adjusting parameter. The bigger r is, the smaller the variance of the normal distribution, and the larger the difference between the target distribution and the actual uniform distribution. We then examine the relative errors of range queries whose ranges are set to be 50. The result of the experiment is shown in figure 3.4(a).

We observe 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 in order to follow a target distribution that is quite different from the actual one.

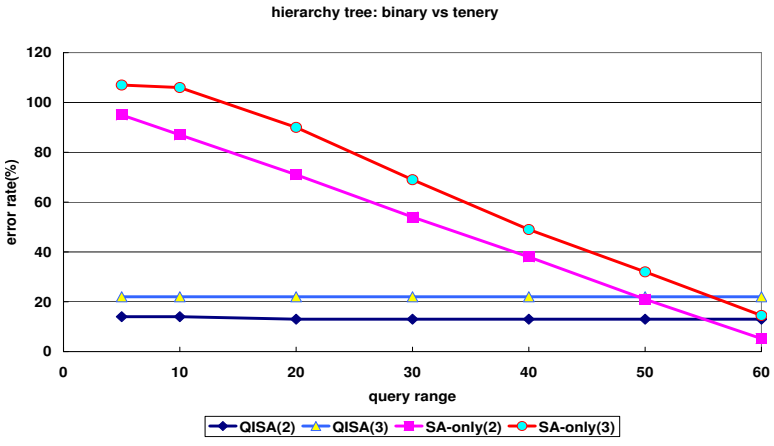
We also investigate the impact when the sensitive attribute values in the original microdata are skewed. We set the actual distribution of the sensitive attribute values to be a zipf distribution. In a zipf distribution, all values in the sensitive attribute domain are ordered and assigned a rank correspondingly. Denote the size of the domain as N. For any element with rank k, its probability is set to be $P(k) = \frac{k^s}{\sum_{i=1}^{N} i^s}$, where s is a control parameter. The smaller the value s is, the less skewed the data is. The control parameter s is set to 0.1 for the source data. The target distribution is also a zipf distribution, with s set to different values. The results are shown in figure 3.4(b).



(a) target:normal



(b) target:zipf



(c) binary vs. ternary

Figure 3.4: Impact of target distribution

From the results we observe that when the target distribution has parameter $s = 0.1$ as the source, we have the best accuracy for query answering. As s increases, the difference between the target distribution and the actual distribution is enlarged, which results in increased relative errors.

Our scheme does not impose any constraints on the structure of the domain generalization hierarchy of the sensitive attribute. The data owner has complete freedom to design their own hierarchy. Since it is infeasible to enumerate all possible hierarchies, we provide some general guidelines. In the following experiment, we choose a balanced binary tree and a balance ternary tree as two domain hierarchies. In the legend of figure 3.4(c), we use (2) to represent the binary hierarchy, and (3) to represent the ternary hierarchy. The sensitive attribute values in the 10k data are uniformly generated in the range [1024,3072]. The target distribution is also set to be a uniform distribution for simplicity.

From the figure we see that the binary hierarchy results in lower relative errors. This is because there are more subdomains in the binary hierarchy than in the ternary one. Thus, statistically we have more choices of smaller subdomains when generating optimal generalizations. Smaller subdomains introduce less uncertainty to the microdata, and thus benefit aggregate query answering.

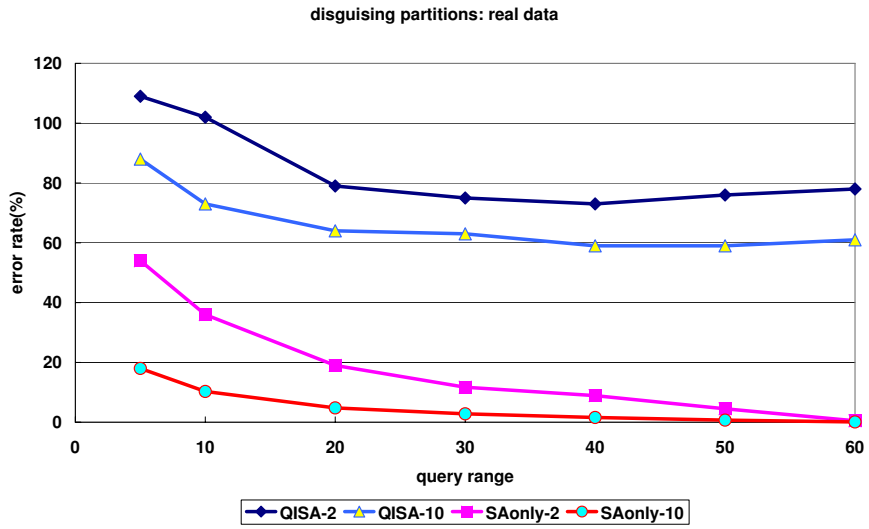
3.7.4 Disguising Distribution for Different Parts of Data

As we have described in section 3.3, one powerful feature of our scheme is that we can disguise the data in a non-uniform way and set different target distributions for different parts of the data. Next we will enforce this mechanism and see how it affects the quality of ad hoc query answering. First, we conduct experiments on the real database. We study two cases in which we partition the data into 2 groups and 10 groups respectively, based on the quasi identifier "age". In each case, the target distribution of each group is set to be the distribution of the source data in that group. In the legend of figure 3.5(a), $-k$ means we partition the table into k groups.

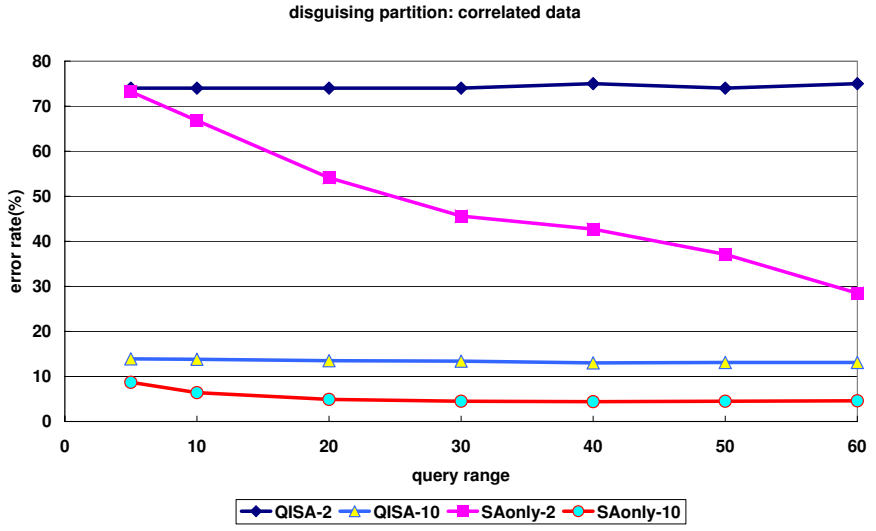
From the figure we see that as the number of partitions increases, the accuracy of both SA-only and QISA improves. The reason is that the number of tuples in each group is getting smaller, and the same range query will cover a bigger portion of each group, which leads to better accuracy of query answering.

Next we use the same correlated synthetic dataset as the one in section 3.7.1. Now we uses a uniform distribution as target for each group. The results are shown in figure 3.5(b).

As previously, we observe the same trend for SA-only and QISA with respect to query



(a) real data



(b) synthetic data

Figure 3.5: Disguising the distributions of different parts of data

range. An additional reason contributing to this result is due to 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. So we can use a smaller hierarchy tree for each group, which leads to smaller generalizations, and reduces the error further. SA-only is still better than QISA, due to the correlation between quasi-identifiers and the sensitive attribute.

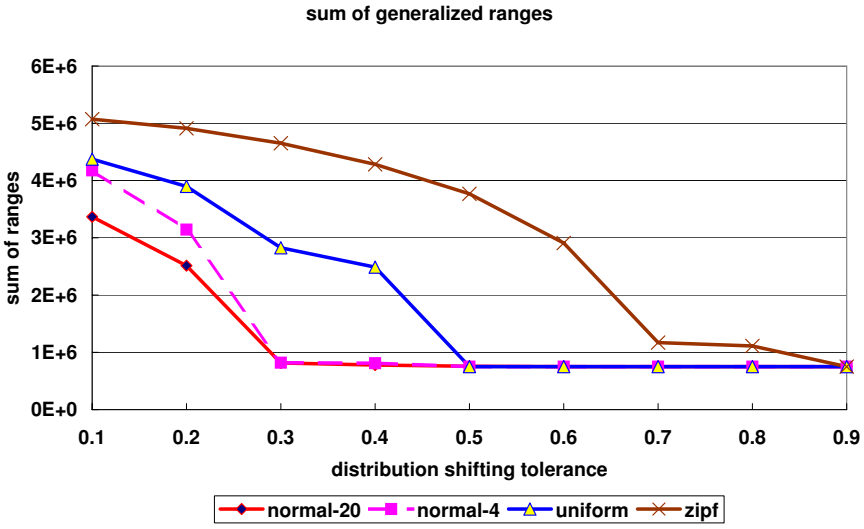
3.7.5 Approximate Target Distribution

As shown in figure 3.4, when the target distribution is far from the source distribution, the query accuracy is poor. In order to improve the query accuracy, data owners may trade off some distributional privacy and specify some privacy tolerance as the shift $x \in [0, 1]$ allowed from the target distribution.

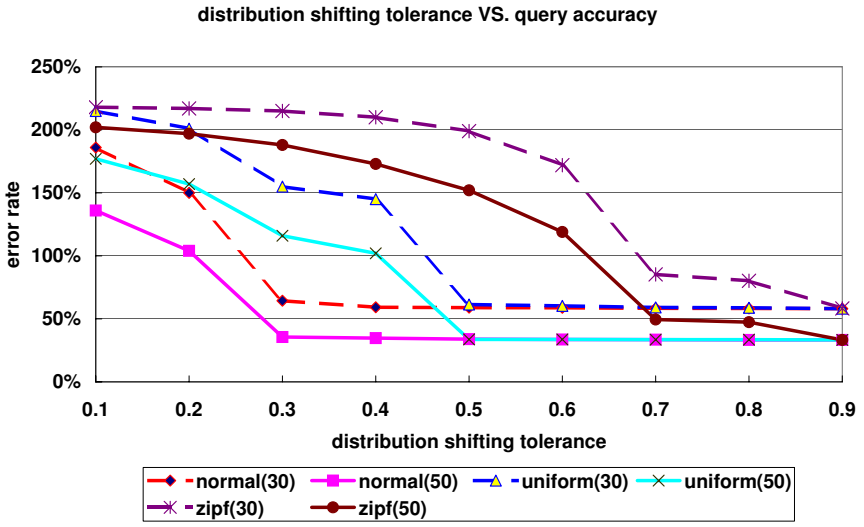
In this experiment, we take the real dataset, and study the tradeoff between the privacy tolerance and query accuracy for different target distributions: uniform, normal with a adjusting parameter 4 and 20 (denoted as normal(4/20) in the legend), and zipf with control parameter 1. First we build the binary privacy-annotated hierarchy S for the source distribution, and normalize the weights(ratios) between each sibling edges to $a_s : b_s$, where $0 \leq a_s \leq 1$ and $a_s + b_s = 1$. For each target distribution, we also construct the corresponding hierarchy H similarly. Then for any tolerance x, we go though each pair of ratios $a : b$ in H, and shift it to $a' : b'$ such that a', b' is closest to the corresponding weights a_s, b_s in S after normalization, and $|a - a'| \leq x, |b - b'| \leq x$. We then generalize the sensitive attribute values on the new hierarchy H'.

First we check the generalized sum of ranges resulting from different target distributions, as shown in figure 3.6(a). We can see that result for normal distribution with adjusting parameter 20 is the smallest, and both normal curves approximate minimum when $x = 0.3$. The uniform distribution goes next, and it approximate minimum at $x = 0.5$. The sum of ranges is the biggest when the target is zipf distribution. For better comparison and understanding, we normalize all the weights to ranges [0-100]. The first two levels of ratios of each distribution are shown in table 3.5.

We can see that the ratios in Level 1 and Level 2 in both Normal(4) and Normal(20) can be transformed into that of the source ratio when $x = 0.3$. Normal(20) is the closest to the source, so its sum of ranges is minimum. In order to transform Level 1 to the source, uniform needs x close to 0.5. So this explains its sum of ranges approximates minimum when $x = 0.5$. Finally, Zipf needs x about 0.8 to transform Level 1 to source, and 0.9 to transform Level 2, that also matches its trends in figure 3.6(a).



(a) Generalized sum of ranges



(b) accuracy vs. privacy tolerance

Figure 3.6: Privacy-accuracy tradeoff

Table 3.5: First two levels of ratios of each distribution

Distribution	Level 1	Level 2	
		left child	right child
source	82:18	3:97	98:2
uniform	50:50	50:50	50:50
normal(4)	50:50	29:71	71:29
normal(20)	50:50	0:100	100:0
zipf	8:92	41:59	9:91

We also run range queries of size 30 and 50. The result is shown in figure 3.6(b). Here we omit the results for uniform(4) to avoid clutter. We can see the trends of query accuracy match those of the sum of ranges perfectly.

3.7.6 Discussion

In our experiments, we do not compare QISA and SA-only with other existing approaches to microdata protection, as no existing techniques have the capability to disguise the actual distribution of parts of the microdata with an arbitrary *target* distribution.

3.8 Summary

In this chapter, we present a generic framework for microdata privacy protection, which allows a microdata owner to specify as the privacy requirement any distribution of the sensitive attribute. The framework then creates a partition of the microdata table. Sensitive attributes of tuples in each group of the partition are generalized and permuted which ensures that even if an attacker uniquely identifies an individual from a tuple's quasi-identifier, the attacker can only derive from the microdata that that individual's sensitive attribute follows the distribution specified by the microdata owner but nothing more.

When the difference between the target distribution and the source distribution is big, the query accuracy is often poor. We further propose two approaches to achieve better accuracy for query answering. The first approach allows data owners to approximate the target distribution in a controlled manner to tradeoff distributional privacy protection and accurate query answering during anonymization. Specifically, we shift the target distribution towards the source distribution within a given threshold, then apply the anonymization algorithm on the new shifted distribution. The

second approach is to inject fake tuples into the anonymized dataset. After data owner specify a threshold for the number of fake tuples, we design optimal algorithms to decide what fake values to be added. Note that although we have injected false information, we can still provide correct deterministic bounds for query answering.

Chapter 4

Anonymizing Bipartite Graph Data Using Safe Groupings

4.1 Motivation

In previous chapters, we have studied the privacy requirements in several different scenarios. In all these scenarios, we have a tabular table, whose attributes are composed of quasi identifiers and sensitive attributes, and we want to protect the privacy of sensitive attributes. However, we further identify that private data can often arise in the form of *associations* between entities. An example is the products bought by customers at a pharmacy. The set of products being sold and their properties is public knowledge, and it may be no secret which customers visit a particular pharmacy. However, the association between a particular individual and a particular medication is often considered sensitive, since it is indicative of a disease or health issue that they have. A large

Table 4.1: Example data set

(a) Author table

Author	Location
a1	US
a2	EU
a3	AS
a4	US
a5	EU
a6	AS

(b) Paper table

Paper	Venue
p1	SIGMOD
p2	VLDB
p3	VLDB
p4	VLDB
p5	SIGMOD
p6	VLDB

(c) Author-Paper table

Author	Paper	Author	Paper
a1	p2	a4	p5
a1	p6	a5	p1
a2	p3	a5	p5
a2	p4	a6	p3
a3	p2	a6	p6
a3	p4		

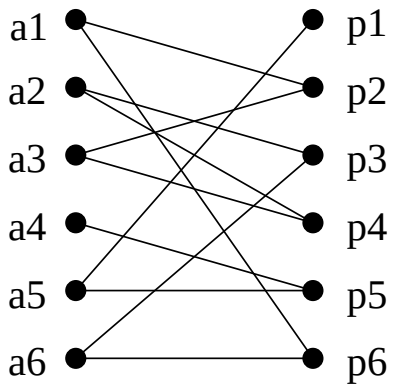


Figure 4.1: Graph representation of example data set

example of association data is the Netflix prize data set, released in 2006, which was anonymized based on an unspecified heuristic method [23]. This led to speculation on how easy it would be to break the privacy [24]. Another example that we shall return to repeatedly is that of authors and papers: for a conference such as SIGMOD, reviewers learn information about submitted papers (title, area, abstract), and could (in future) also see detailed information about authors who have submitted papers, in order to verify conflicts of interest. But, since SIGMOD is a double-blind conference, the association between authors and papers should not be revealed to reviewers.

The most natural way to model such data is as a graph structure: nodes represent entities, and edges indicate an association between them. In this work, we study data that can be modeled as bipartite graphs—there are two types of entity, and associations link together one entity of each type. In the pharmacy, customers buy products, and in SIGMOD, authors write papers, building (customer, product) and (author, paper) associations. Each entity can be involved in few or many associations, but in most common situations, only a tiny fraction of all possible associations are present. In other words, the induced graph is quite sparse, and we must ensure that these associations are not easily revealed.

Although the data is private, it is still desirable to allow aggregate analysis. Pharmaceutical companies wish to understand which pattern of products are bought by people in particular age ranges; public health organizations want to watch for disease outbreaks affecting particular demographics based on certain types of medicine being purchased; SIGMOD may encourage analysis of hot topics in databases, or better understanding of coauthorship patterns. Publishing the raw data would allow these questions to be answered directly, but would fail to meet the privacy concerns

outlined above. The model where the data owner accepts queries and either adds noise to results or refuses to answer some questions requires the data owner to be an active participant and may limit what analysis is possible. Instead, we adopt the approach of publishing some anonymized version of the data, and ensuring that the scope for inferring any given association from this data is limited. This approach allows a wide variety of ad hoc analyses and novel valid uses of the data, while ensuring our privacy goals are met.

The problem of publishing anonymized data has attracted significant interest in recent years [7, 3, 25, 2, 9, 21, 22]. However, the focus has mostly been on tabular data, rather than the associations we study here. As a consequence, applying existing anonymization techniques tends to erase almost all structure, so that little use can be made of the resulting data. Moreover, a tabular approach ignores the inherent graph properties which hold a lot of the value of the data: e.g. structure such as number of authors of a paper co-authorship, collaboration "hop" distance between a pair of authors, pattern of collaboration between authors on a paper, and so on. These are all important features of interest for aggregate analysis, but are radically altered by simply treating the data as a table and masking or perturbing the data. In Section 4.3, we work through several detailed examples to show that existing approaches for tabular data are insufficient for handling associations.

Some recent work has begun to address anonymizing graph data, motivated by social network structure. But rather than proposing ways to ensure privacy, such work has tended to analyze the strength of attacks on simple anonymization schemes [4, 26]. These works tend to assume a large amount of knowledge or power on behalf of the attacker, and show that under such assumptions some associations can be inferred. In contrast, we give a new approach for anonymizing associations which can be represented as bipartite graphs and show it to be resilient against a variety of attacks.

Our Contributions. Our methodology is based on the idea that rather than masking or altering the graph structure, we should preserve the graph structure exactly, and instead focus on masking the mapping from entities to nodes of the graph. This approach ensures that the complex and sensitive graph structure is not affected, and so we can be sure that any analysis based principally on the graph structure will be correct. Privacy is ensured by *grouping* the nodes and entities: we partition the nodes in the graph, and the corresponding entities into groups so that, given a group of nodes, there is a (secret) mapping from these nodes to the corresponding group of entities. There is no information published that would allow an attacker to work out, within a group, which nodes corresponds to which entity. This gives a tradeoff between privacy and utility: intuitively, larger

groups give more privacy, but less certainty when answering queries which select a subset of entities.

We give a simple condition for a grouping to be *safe*, which precisely limits the ability of an attacker to make any inference from the published information alone. We provide an algorithm which is successful at finding safe groupings in practice, and go on to describe how to answer a variety of query types efficiently given the published anonymized data. We also give formal analysis of how little can be deduced by an attacker who has additional background knowledge in the form of known associations between particular pairs of entities, and show that there is high security for entities about whom no information is known by the attacker.

We demonstrate the efficacy of our approach with a careful experimental analysis of the ease of building safe groupings, and the accuracy with which a variety of queries can be answered over such anonymized data. We also study the effect of variations of our approach, and demonstrate that techniques based on publishing two versions of the same data, while significantly increasing the utility and accuracy of query answering, can also expose more associations to unintended revelation.

4.2 Preliminaries

4.2.1 Graph Model

Throughout, we focus on problems of anonymizing bipartite graphs $G = (V, W, E)$ (bigraph for short). That is, the bigraph G consists of $m = |V|$ nodes of one type, $n = |W|$ nodes of a second type, and a set of $|E|$ edges $E \subseteq V \times W$. Such graphs can encode a large variety of data, in particular, the set of existing links between two sets of objects. For example, we can encode which products at a supermarket were bought by a set of customers; which websites were visited by users; which courses were taken by students; and so on. Throughout, we shall work with a single familiar illustrative example, that of a set of authors $A = V$ and a set of papers $P = W$. An edge (a, p) indicates author $a \in A$ was an author of paper $p \in P$. Observe that here, as in many of the examples above, the graph is relatively *sparse*. That is, each author typically writes only a small fraction of all papers, and each paper is written by only a few authors. As a consequence, the number of edges e is small compared to the number of *possible* edges, which is $n \times m$. In full generality, we can consider graphs with multiedges, with weights or additional attributes. However, for clarity, we describe only the unweighted, undirected, single edge case: this has sufficient richness to capture many challenging problems.

In a relational database, a bipartite graph $G = (V, W, E)$ is naturally and concisely rep-

resented by three tables, corresponding to V, W and E. In our example, we would have a table of authors V, including attributes such as affiliation and location; a table of papers W, including attributes such as title, venue, and date; and an author-paper table E encoding who wrote what. Thus *entities* in the tables V and W correspond to *nodes* in the graph defined by E, in a one-to-one fashion.

Example 1. *Table 4.1 shows a sample instantiation of this schema with Figure 4.1 showing the graph representation of the author-paper relation in Table 4.1(c). Authors have an additional attribute, location, indicating whether they are based in the US, Europe (EU) or Asia (AS). The venue of a paper indicates which conference it was published in. Since the graph accurately represents the relational data, we use both graph and relational terminology.* □

4.2.2 Privacy Goals

Our objective is to publish an anonymized version of the graph G, which still allows a broad class of queries to be answered accurately, but which maintains privacy of the associations. To make this goal precise, we describe our privacy goals, and outline classes of queries which we aim to answer.

Our privacy objective is based on the idea that in many cases it is the *association* between two nodes which is private and must be disguised. As noted, the set of customers of a pharmacy may not be considered particularly sensitive, and the set of products which it sells may be considered public knowledge. However, the set of products bought by a particular customer is considered private, and should not be revealed. For now, we focus solely on preserving the privacy of associations, and assume that properties solely of entities (e.g. venue of a paper) are public. Clearly, there are situations with differing privacy requirements, which we comment on in Section 4.6.

Since it is desirable to allow answering of ad hoc aggregate queries over the data (e.g. how many customers from a particular zip code buy cold remedies), we wish to release some anonymized version of this data which gives accurate answers to such queries but protects the individual associations. More strongly, we want the graph properties of the data to be preserved. This corresponds to simple features, such as the degree distribution of the nodes, but also more long-distance properties, such as the distribution of nodes reachable within two steps, three steps, etc.

Here, as in all work on anonymization, there is an inherent tradeoff between *privacy* and *utility*, although this can be hard to quantify precisely. Various extreme approaches maximize one over another: publishing the original data unchanged clearly maximizes utility, but offers no privacy;

removing all identifying information and publishing only an unlabeled ("fully censored") graph gives high privacy, but limited utility for aggregate queries over nodes satisfying certain predicates.

Prior work has considered strong dynamic attack models (where nodes and edges can be inserted into the graph), which can result in some small number of associations being revealed [4]. We argue that for the situations we consider, this is not a realistic model. It assumes that an attacker knows what data will be covered by the release and can easily modify it in advance. But, in the pharmacy example, adding edges means particular individuals must buy certain products in certain stores at certain times, which may be remote from the location of the attacker. Adding nodes would involve creating new products for sale in the stores, which seems most implausible. Even then, such attacks [4] only reveal information about entities for which some information is already known. Entities not involved in the attack remain secure in their work.

In extreme cases, the graph structure can still leak information about individual edges: for example, if the underlying graph is complete then we can deduce the existence of any edge from the censored graph structure alone. In typical cases such as the examples we consider, virtually nothing can be deduced from the graph structure alone. We aim for privacy guarantees relative to the baseline of the unlabeled graph. In particular, we want to study what guarantees can be made in the following scenarios:

Definition 4.2.1. *In the* static attack *case, the attacker analyzes solely the information which is published by the scheme, and tries to deduce explicit associations from this information. Ideally, the number of associations which can be correctly inferred (beyond what is implicit in the censored graph) should be minimal if not zero.*

In the learned link attack *case, the attacker may learn some associations — for example, that author a1 wrote paper p2. Again, ideally the additional associations that can be inferred should be minimal if not zero, when the number of link revelations is small.*

We are principally concerned with an attacker being able to make *positive* inferences, e.g. being able to deduce that a6 wrote p6. We are less concerned about *negative* inferences, e.g. deducing that a1 did not write p1. Since the graphs we are concerned with are sparse, and the maximum degree is small compared to the total number of nodes, we consider such discovery to be entirely acceptable (the same assumption is implicitly present in much of the prior work on anonymizing tabular data). We should only be concerned when a sequence of negative inferences eliminates enough possibilities to leave a positive inference: learning that a4 wrote at least one paper but did not write p1, p2, p3, p4, or p6 allows us to infer that a4 did write p5. However, we treat this

as an instance of the positive inferences that we seek to limit.

4.2.3 Query Types

As in prior work, given the difficulty of giving a precise privacy/utility tradeoff, we consider approaches which first fix a given level of privacy and then try to optimize and measure the utility, or vice-versa. In order to more precisely analyze utility, we describe a set of sample aggregate query types which we wish to support. These can be based on predicates over solely graph properties of nodes (such as degree), which we denote P_n, and predicates over attributes of the entities, P_a. In our author-paper example, P_a could select out authors from the US, or papers published in VLDB, while a typical P_n might be that a paper has a single author. We separate these two types of predicates, since when we publish a censored graph, we can still evaluate P_n predicates exactly, whereas we have complete uncertainty in applying P_a predicates.

We list a set of types of queries of increasing complexity, based on standard SQL aggregates (sum, count, avg, min, max):

- Type 0—Graph structure only: Compute an aggregate over all neighbors of nodes in V that satisfying some P_n.

E.g.: Find the average number of papers per author;
Compute the average number of single author papers per author.

- Type 1—Attribute predicate on one side only: Compute an aggregate for nodes in V satisfying P_a; Compute an aggregate on edges to nodes in V satisfying P_n from nodes in W satisfying P_a.

E.g.: Find the average number of papers for US authors;
Count the total number of single author papers of US authors.

- Type 2—Attribute predicate on both sides: Compute an aggregate for nodes in V satisfying P_a to nodes in W satisfying P'_a.

E.g.: Count the total number of papers by US authors in SIGMOD; Find the most single author VLDB papers by a US author.

Naturally, one can define yet higher orders of queries that are more complex, either through more constraints or more steps through the graph. Join-style queries would compute an aggregate of nodes from V at distance 2 from nodes in V satisfying P_a connected via nodes in W satisfying P'_a, and so on. One can also bring in other graph properties (such as measuring the diameter of an induced subgraph). For this work, we constrain our interest principally to the classes

of queries defined above, since these are sufficiently rich to be challenging to answer accurately, while being sufficiently concise to specify compactly and work with over realistic data sets. In particular, note that while queries of type 0 can easily be answered on the fully censored graph exactly, answering queries of other types requires some more information about attributes of the entities in order to give any reasonable accuracy.

Datasets and Experimental Environment. All experiments for this paper were implemented in JDBC and SQL Server 2000. The primary dataset used is derived from DBLP and corresponds to all conference data. It was retrieved from `http://dblp.uni-trier.de/xml/` on 06/21/2007. The data set contains $|V| = 402023$ distinct authors, $|W| = 543065$ distinct papers, and $|E| = 1401349$ (author, paper) edges. We also carried out experiments on association data from other sources such as the Internet Movie Database (actors, movies), but in the interest of brevity, we only present results from DBLP, since other data gave similar conclusions.

4.3 Applying Existing Techniques

A natural first approach to addressing these privacy questions is to apply prior work on table anonymization, since we have observed that tables can represent graph data. However, such prior anonymization techniques only try to preserve the accuracy of table-based queries, and do not consider any graph semantics. As a result, we show that fundamental graph properties are quickly lost under such transformations.

4.3.1 Representing as a relation

Representing the author-paper example in Table 4.1 using tables, gives an author relation (Table 4.1(a)), a paper relation (Table 4.1(b)), and an author-paper relation (Table 4.1(c)). We can join these to make a single table (Table 4.2(a)), and try to anonymize it. In our example, each row lists an author, a paper, the author's location, and the paper's venue. How can we meet our goal of not revealing any (author, paper) association by applying a k-anonymization algorithm? Removing all author IDs destroys all graph structure. Setting author as a quasi-identifier and paper as sensitive attribute fails because k-anonymization allows k papers of the same author to be grouped together, since they share a quasi-identifier. Setting (author, paper) as the sensitive attribute fails, because k-anonymization does not alter or mask sensitive attributes. Instead, we could add a dummy sensitive attribute of "true" to each row to indicate that the association is sensitive. The k-anonymized version

Table 4.2: Attempting to apply existing anonymization to graph data

(a) Original data table

Author	Paper	Location	Venue
a1	p2	US	VLDB
a1	p6	US	VLDB
a2	p3	EU	VLDB
a2	p4	EU	VLDB
a3	p2	AS	VLDB
a3	p4	AS	VLDB
a4	p5	US	SIGMOD
a5	p1	EU	SIGMOD
a5	p5	EU	SIGMOD
a6	p3	AS	VLDB
a6	p6	AS	VLDB

(b) 3-anonymous data table

Author	Paper	Location	Venue
*	*	*	VLDB
*	*	*	VLDB
*	*	*	VLDB
*	*	*	VLDB
*	*	AS	VLDB
*	*	AS	VLDB
*	*	*	SIGMOD
*	*	*	SIGMOD
*	*	*	SIGMOD
*	*	AS	VLDB
*	*	AS	VLDB

(c) Matrix (top), 3-anonymized (bottom)

	p1	p2	p3	p4	p5	p6
a1	0	1	0	0	0	1
a2	0	0	1	1	0	0
a3	0	1	0	1	0	0
a4	0	0	0	0	1	0
a5	1	0	0	0	1	0
a6	0	0	1	0	0	1

	p1	p2	p3	p4	p5	p6
a1	*	*	0	*	*	*
a2	0	0	*	*	*	*
a3	*	*	0	*	*	*
a4	0	0	*	*	*	*
a5	*	*	0	*	*	*
a6	0	0	*	*	*	*

of this table must use generalization and suppression to ensure each row is indistinguishable from $k-1$ others [20, 2]. Options for concealing author and paper identifiers are limited: since they are arbitrary identifiers, there is no natural hierarchy for generalization so they can only be withheld. A 3-anonymized version of our example data set is shown in Table 4.2(b), which provides very low utility.

This attempt at anonymization loses the notion of individual authors and papers, and so is unable to give useful answers to the query types outlined above. One could augment the anonymized data with additional information, but this risks breaching privacy and does not guarantee to anticipate all reasonable queries which could be formulated: recall that the purpose of publishing anonymized data is to allow a broad variety of ad hoc queries to be posed.

4.3.2 Representing as a matrix

A fundamental problem with the above approach is that k-anonymity is formally defined so that there should be at least k *individuals* whose representation is identical; in this representation, each individual is present in multiple places, so for example in Table 4.2(b), two rows in the anonymized table refer to the same author, giving them weaker privacy. This leads us to represent the graph data instead as a binary matrix: rows correspond to nodes in V, columns to nodes in W, and an entry (i, j) is set to 1 if there is an edge between $v_i \in V$ and $w_j \in W$, and 0 otherwise. We can now take such a table, and try to apply existing anonymization techniques on it. Similar to above, the only meaningful anonymization of a 0 or 1 value is to generalize to "*".

Applying k-anonymization is similar to having author as a quasi-identifier and paper as a sensitive attribute: now papers with more than k authors may be revealed, while other papers may be fully masked. This also virtually wipes out the utility of the data. For example, Table 4.2(c) shows the matrix representation of the sample data from Table 4.1, and the result of 3-anonymizing it: all that is left is a few negative associations. The fundamental problem here is that these approaches have two equally unpalatable options: either an association is fully revealed, or else it is withheld.

4.3.3 Anonymization Through Permutation

A third approach to anonymizing tabular data is based on the idea of "permutation": breaking the links between quasi-identifiable attributes and sensitive attributes [21, 22]. This seems more suited to the graph setting: we have an association between nodes in a graph that we wish to anonymize. This leads to the following algorithm: form edges into groups, and within each group,

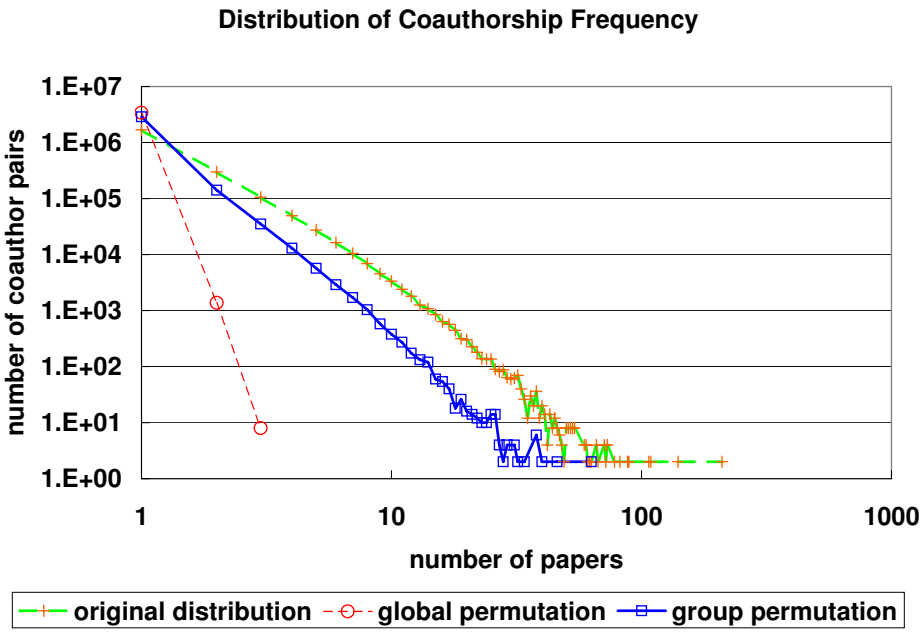


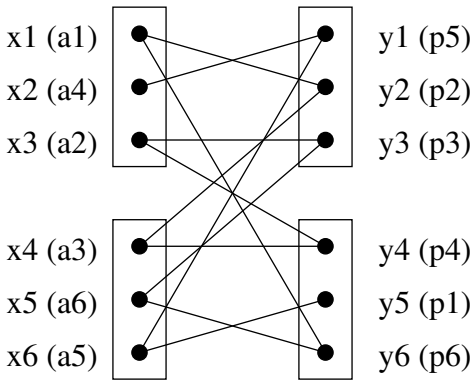
Figure 4.2: Permutation destroys the co-author relationship

publish the pair of node (multi)sets that form edges. For example, grouping the edges from Figure 4.1 into sets of size 3 and 4 based on author pairs gives:

$$(\{a1, a1, a2, a2\}, \quad \{p2, p3, p4, p6\}),$$
$$(\{a3, a3, a4\}, \quad \{p2, p4, p5\}),$$
$$(\{a5, a5, a6, a6\}, \quad \{p1, p3, p5, p6\})$$

Equivalently, for a group containing edges $e_1 = (v_1, w_1), e_2 \ldots e_\ell$ we generate ℓ permuted edges by picking a random permutation π and publishing $e'_1 = (v_1, w_{\pi(1)}) \ldots e'_i = (v_i, w_{\pi(i)}) \ldots e'_\ell$. Conceptually, imagine taking every edge in the group and "breaking it in the middle", then forming new edges by joining half-edges from V to half-edges from W. This method initially seems more promising than the above, since it guarantees to preserve node degrees (i.e. the number of papers linked to an author is the same before and after the permutation, and vice-verse), and the true mapping from authors to papers seems well-masked. However, when we try to evaluate simple graph queries (type 0) over this data, we find that the results are highly inaccurate.

We created a global permutation of the DBLP data where all edges are placed into a single group and permuted. We also created a small group permutation based on sorting papers primarily by conference and year and then by author count. Small groups were formed from edges corresponding to pairs of papers. Figure 4.2 shows the result of plotting the distribution of the frequency



(a) Grouped and Relabeled Graph

x1	y2
x1	y6
x2	y1
x3	y3
x3	y4
x4	y2
x4	y4
x5	y3
x5	y6
x6	y1
x6	y5

E'

Customer	Group
c1	CG1
c2	CG1
c3	CG2
c4	CG1
c5	CG2
c6	CG2

H_V

Product	Group
p1	PG2
p2	PG1
p3	PG1
p4	PG2
p5	PG1
p6	PG2

H_W

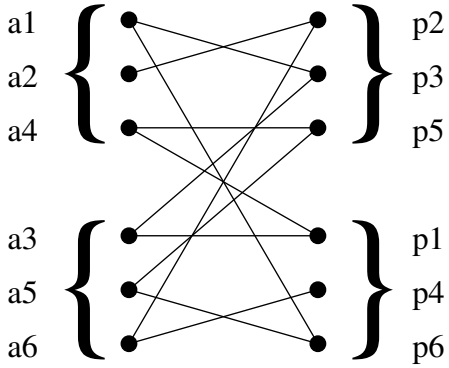
X-node	Group
x1	CG1
x2	CG1
x3	CG1
x4	CG2
x5	CG2
x6	CG2

R_V

Y-node	Group
y1	PG1
y2	PG1
y3	PG1
y4	PG2
y5	PG2
y6	PG2

R_W

(b) Published tables representing $(3, 3)$ grouped graph



(c) Graph of Published Data

Figure 4.3: (3,3)-anonymization of example relation

of co-authorship for each pair of co-authors. The results clearly demonstrate that this permutation approach does not accurately maintain the coauthor relationship. In the source data 1.6M pairs of coauthors have written at least one paper together, and one pair has co-written 210 papers. In the global permutation, the maximum number of papers coauthored together is only three. The permutation of small groups is closer to the source distribution, but the error is still significant. This is unsurprising since coauthors often collaborate over long periods, writing multiple papers together. Permutation of papers breaks this correlation and links unrelated authors. All other grouping methods that we tried similarly failed to preserve these basic graph properties.

4.4 Privacy through Grouping

All the above attempts to use existing techniques render the data virtually unusable for the simple reason that they alter or mask the graph structure in ways that fundamentally alter its properties. In contrast to the table case, where modifying a row has relatively minor impact on table properties, adding or deleting an edge can have significant impact on properties of a graph (for example, it can change a graph from being connected to disconnected). So techniques which involve perturbing the graph structure, for example by redirecting edges, will be considered unacceptable. Instead, we focus on techniques which retain the entire graph structure but perturb the *mapping from entities to nodes*. That is, methods that publish a set of edges E' that are isomorphic to the original edges E, but where the mapping from E to E' is partially or fully masked.

4.4.1 Definition of Grouping

In this paper, we focus on masking the mapping via *grouping* the nodes of the graph. This technique preserves the underlying graph structure perfectly, but masks the exact mapping from entities to nodes. We provide some formal definitions.

Definition 4.4.1. *Given a set V, a k-*grouping *is a function H such that for any $v \in V$, the subset $V_v = \{v_i \in V : H(v_i) = H(v)\}$ has $|V_v| \geq k$. Formally,*

$$\forall v \in V : \exists V_v \subseteq V : |V_v| \geq k \wedge (\forall v_i \in V_v : H(v_i) = H(v))$$

That is, H partitions of V into subsets of size at least k. The k-grouping is strict *if every group V_v has size* exactly *k or $k+1$.*

In other words, a k-grouping partitions V into non-intersecting subsets of size at least k. The strictness property insists all groups in a k-grouping be close to k in size, since smaller groups

allow more accurate query answering. k-grouping ensures that each group is at least size k, to meet privacy goals; *strictness* ensures no group is much larger than k, for accuracy. We use the definition of grouping to publish a modified version of the graph:

Definition 4.4.2. *Let F_V be a relabeling function to relabel elements of V injectively onto a new set $F(V)$; and let F_W be a relabeling function to relabel elements of W injectively onto a (disjoint) set $F(W)$. Given a k-grouping on V, H_V, and an ℓ-grouping on W, H_W, of a graph $G = (V, W, E)$, define the (k, ℓ)-*grouped graph *G' as $G' = (V, W, H_V, H_W, E', R_V, R_W)$ where:*
(a) V and W are the original sets of entities V and W, and H_V and H_W are the grouping functions defined above.
(b) E' is the relabeled edge set given by

$$E' = \{(F_V(v), F_W(w))|(v, w) \in E\}.$$

(c) R_V, R_W are remappings defined by

$$R_V(F_V(v \in V)) = H_V(v) \text{ and } R_W(F_W(w \in W)) = H_W(w).$$

When both H_V and H_W are strict, this is a strict (k, ℓ)-grouping.

Example 2. *For the example in Table 4.1, set groups AG1, AG2 (author group 1 and 2) and PG1, PG2 (paper group 1 and 2) as*

$$H_A^{-1}(AG1) = \{a1, a2, a4\} \quad H_A^{-1}(AG2) = \{a3, a5, a6\}$$
$$H_P^{-1}(PG1) = \{p2, p3, p5\} \quad H_P^{-1}(PG2) = \{p1, p4, p6\}$$

This is a strict $(3, 3)$-grouping since every author group and every paper group has (exactly) three members. The resulting grouped graph is shown in Figure 4.3(a), with the arbitrary relabeling of nodes on xi's and yi's. The published information can be derived from this: Figure 4.3(b) shows the five published tables (in addition to the original author and paper tables, Table 4.1(a) and 4.1(b)). The result is also compactly represented as a graph in Figure 4.3(c): this shows the underlying edge structure, and the fact that a set of nodes map to a set of entities, but the exact mapping is not revealed. □

The key properties of this modified graph as a function of the original graph G is given by the choice of the grouping functions H_V and H_W. This definition is well-suited to storage within a relational database. In the above example, we publish author and paper relations as before (corresponding to V and W); author-group and paper-group tables which encode the mapping of each author and paper to groups (corresponding to H_V and H_W); a masked-author-paper relation, in which each author and paper is mapped to a new node id (corresponding to E'); and lastly masked-author-group and masked-paper-group tables which map from the masked identifiers to

groups (corresponding to R_V and R_W). It is important that the base relations corresponding to V and W do not contain any information relating to the graph, such as the degree of the node. Otherwise, an attacker could potentially use this to relink between rows of V or W and nodes in E'. We give two further examples which illustrate extremes of privacy and utility

Example 3. Perfect Utility. *Setting $H_V(v) = v$ and $H_W(w) = w$, (the identity functions) gives a $(1, 1)$-grouped graph G'. Here, $E' = E$, and hence G' encodes the original graph G exactly. Every query on G' can be answered with the same accuracy as on G. So there is perfect utility, but no more privacy than we began with.* □

Example 4. Perfect Privacy. *Setting H_V to map all $m = |V|$ members of V to the same group, say 0, and H_W to map all $n = |W|$ members of W to, say, group 1 gives the (m, n)-grouped graph G'. G' has no useful information mapping between entities in V, W and the nodeset of E'. That is, we publish entity tables and the fully censored graph. Recall we stated that in our model, it is acceptable to publish a censored graph, and so we say that this grouping guarantees privacy. This retains the graph structure, as required, but completely removes the mapping from entities (e.g. authors and papers) to nodes in the graph. We cannot have any more privacy in our setting, when we insist on publishing at least this much information. This offers very limited utility in answering the query types listed in Section 4.2.3, since we cannot apply any selective attribute predicate with any certainty.* □

Between these two extremes lie many possibilities which trade off utility and privacy. Given (k, ℓ)-grouped graph, where both k and ℓ are fairly small, aggregate queries such as those described in Section 4.2.3 can be answered approximately. Bounds can be placed on the answers within which the true answer must fall. When k and ℓ are small, these bounds will be narrow; as k and ℓ grow large, the bounds will widen accordingly. Thus, choices of k and ℓ define a lattice over possible groupings, bounded by $(1, 1)$ and (m, n). This is illustrated in Figure 4.4, which includes points of note that are defined and discussed in subsequent sections.

4.4.2 Safe Groupings

There are many ways to form a k-grouping, but not all of these offer the same level of privacy, due to the local graph structure.

Example 5. *Consider a large graph G, which happens to contain the complete subgraph between nodes $\{v_1, v_2, v_3\}$ and $\{w_1, w_2, w_3\}$. Suppose $\{v_1, v_2, v_3\}$ forms the entirety of one group in H_V,*

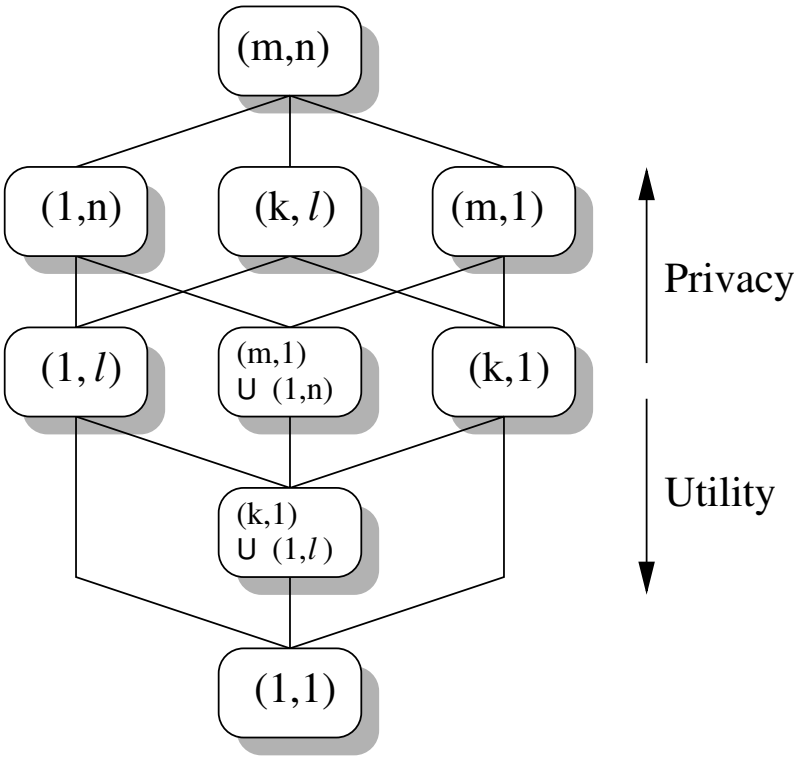


Figure 4.4: Lattice over groupings and privacy/utility tradeoff

and $\{w_1, w_2, w_3\}$ forms the entirety of a group in H_W. From the published G', it is possible to infer immediately all the connections between these six nodes (a static attack). Such inference is not possible on the fully censored version of G', but the unfortunate choice of grouping allows information to leak. □

Some natural attempts to fix this, such as insisting that the density of edges between any pair of groups is low, fails to preserve the same level of privacy as edges are learned by an adversary. We define a notion of 'safe grouping', which we subsequently prove is more robust against static and learned link attacks.

Definition 4.4.3. *H_V is a* safe *grouping of V in the context of a graph $G = (V, W, E)$, if the following condition holds:*

$$\forall v_i \neq v_j \in V : \quad H_V(v_i) = H_V(v_j) \Rightarrow \nexists w \in W : (v_i, w) \in E \wedge (v_j, w) \in E$$

By extension, a (k, ℓ)-grouping of a graph G is safe if H_V and H_W are both safe groupings.

That is, a safe grouping ensures that any two nodes in the same group of V have no common neighbors in W (the definition for a safe grouping of W is symmetric, interchanging the roles of V and W). In the author-papers example, it means that no two authors in the same group have co-authored any papers if the grouping is safe. Given G and $k > 1$, there is no guarantee that there exists a safe k-grouping (for $k = 1$, all groupings are trivially safe).

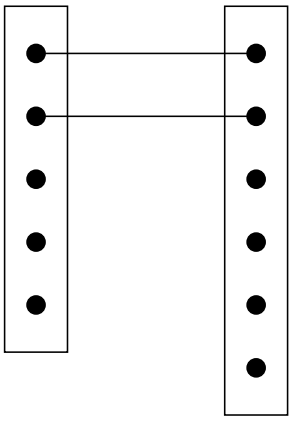
A necessary condition for the existence of a safe (k, ℓ)-grouping arises from the *sparsity* of the graph. A group of size k in V and a group of size ℓ together induce a subgraph of G which could have at most $k\ell$ edges. However, if the grouping is safe then (within the induced subgraph) any node can have degree at most 1; otherwise, there are two nodes with a common neighbor. Figure 4.5(a) shows a typical structure between two groups of size $k = 5$ and $\ell = 6$. So there can be at most $\min(k, \ell)$ edges between these two groups. This is true for every possible pair of groups. Since every edge touches exactly two groups, the density of the graph, defined by $|E|/|V||W|$, can be at most $\min(k, \ell)/(k\ell) = 1/\max(k, \ell)$. Finding a k-grouping can be hard even for small values of k, by the below theorem:

Theorem 6. *Finding a safe, strict 3-grouping is NP-hard.*

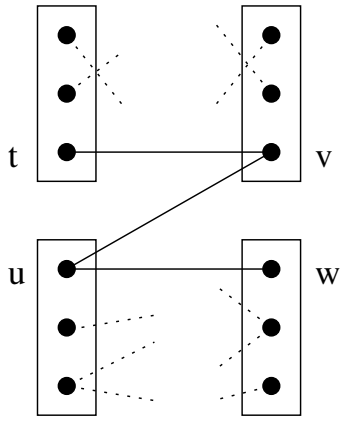
Proof. Define $G^2(V) = (V, E^2)$ as the (non-bipartite) graph on V so that

$$(v_i, v_j) \in E^2 \iff \nexists w \in W : (v_i, w) \in E \wedge (v_j, w) \in E.$$

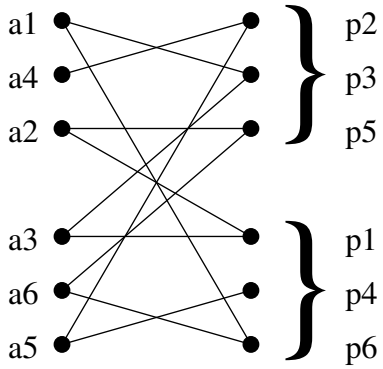
The requirement on H_V to be safe is equivalent to requiring every pair of nodes in the same group form an edge in E^2. That is, the group of nodes in the grouping forms a clique in $G^2(V)$. Therefore,



(a) Structure of safe groups



(b) (t, v) and (u, w) imply (u, v)



(c) Safe $(1, 3)$-grouped graph

Figure 4.5: Safe Grouping Examples

Algorithm 5 $Group(V, W, E, k) j \leftarrow 0$

```
while |V| ≠ 0 do
  for u ∈ V do
    i ← 1;
    while (∃v ∈ VG_i, w ∈ W : (v, w) ∈ E ∧ (u, w) ∈ E) ∨ |VG_i| > k + j do
      i ← i + 1;
    end while
    VG_i ← VG_i ∪ u;
  end for
  j ← j + 1; V ← ∅; i ← 1; l ← 1;
  for i : (|VG_{i(j−1)}| > 0) do
    if |VG_{i(j−1)}| ≥ k + j then
      VG_{lj} ← VG_{i(j−1)}; l ← l + 1;
    else
      V ← V ∪ VG_{i(j−1)};
    end if
  end for
end while
```

a strict 3-grouping of V corresponds to a partition of $G^2(V)$ into triangles (when $m = |V|$ is a multiple of 3). For any desired graph $G_1 = (V_1, E_1)$, define a bigraph $G = (V, W, E)$ such that $G^2(V) = G_1$: create $V = V_1$ and $W \subseteq V \times V$, and for each $(v_i, v_j) \in E_1$, insert $(v_i, (v_i, v_j))$ and $(v_j, (v_i, v_j))$ into E. Since partitioning a graph into triangles is NP-hard (problem [GT11] in [27]), and we can encode this problem as an instance of finding a safe, strict 3-grouping, we conclude that this problem is NP-hard also. □

However, safe groupings can be found easily when the graph is sparse. For a bigraph $G = (V, W, E)$ where every node has degree 1 (i.e. E gives a matching between V and W), every possible grouping is safe, trivially. More generally, when the graph is sparse and has few high degree nodes, safe k-groupings can be found for most moderate values of k. The intuition is that the constraints posed by the edges of the graph are easy to satisfy when not too many edges are present.

Most of the graph types discussed already are quite sparse and have few nodes of high degree: most shoppers purchase only a small number of the items on sale in a store, and most items

are purchased by a fraction of all shoppers; most authors write only a small number of papers relative to the total number of papers written, and most papers have a small number of authors. Studying the data from DBLP, we observe that the most prolific author has written around 400 papers (out of 500K), and most authors on a single paper is about 100 (out of 400K). In total, there are only 1.4M edges in the author-paper graph, out of a possible $400K \times 500K = 200,000M$, making this very sparse. Section 4.4.5, gives an algorithm to find a safe grouping.

4.4.3 Security of (k, ℓ)-Groupings

We analyze what can be deduced by an attacker presented with a safe (k, ℓ)-grouping of graph data, where both k and ℓ are greater than 1. We first argue that safe groupings are secure against the static attacks defined in Definition 4.2.1.

Lemma 4.4.1. *In a safe grouping, given nodes $v \in V$ and $w \in W$ in groups of size k and ℓ respectively, there are $k\ell$ possible identifications of entities with nodes and the edge (v, w) is in at most a $1/\max(k, \ell)$ fraction of such possible identifications.*

Proof. Consider a group VG of V containing k nodes, and a group WG of W containing ℓ nodes. In the subgraph of G induced by VG and WG, there are $0 \leq e \leq \min(k, \ell)$ edges, following from the definition of safe grouping. There is no information available in what is published to break the symmetry between the nodes of VG, or between the nodes of WG. Recall, we insist that tables V and W contain no data related to the graph itself, such as degree or neighborhood, that could break this symmetry.

For any entities $v \in VG$ and $w \in WG$, it is feasible that (v, w) is an edge, and also feasible that (v, w) is not an edge. More strongly, consider the number of ways of identifying entities v and w with the anonymized nodes $\{x1 \dots xk\}$ and $\{y1 \dots y\ell\}$. Since all $k\ell$ possibilities are feasible, then there is an edge between v and w in exactly an $e/k\ell$ fraction of feasible configurations, i.e. at most $\min(k, \ell)/k\ell = 1/\max(k, \ell)$, the bound on the density of the whole graph derived in Section 4.4.2. Since this analysis holds for every pair of groups, then the attacker cannot infer any associations with certainty. □

Under this measure, a $(k, 1)$ grouping offers the same static guarantee as a (k, k)-grouping. However, as we discuss in more detail in Section 4.4.6, there are other factors to consider. We remark on a connection to the concept of ℓ-diversity [3]: here, the requirement is that between two groups the fraction of sensitive information (associations that are present) is bounded by $1/\max(k, \ell)$,

which is similar to the ℓ-diversity requirement. If there are small groups, the attacker's confidence in a particular association can be higher. In particular, two groups of size 1 with an edge between them corresponds to a known association between entities. Although a safe (k, ℓ)-grouping has no groups of size 1, in the active attack model, when an attacker learns the existence of an edge (v, w), he may be able to refine the grouping in order to create groups of size 1. We will show that this refinement has bounded impact on the security of entities not directly impacted by the edge revelation, after presenting an example where an attacker may learn an association.

Example 6. *Consider the four groups shown in Figure 4.5(b), and the three edges that connect them. Other nodes in the same groups have edges to other groups (dashed lines) which do not affect this example. In the static case, as proved above, the attacker cannot make any strong inferences. However, in the link learning case, if the attacker learns (t, v) is an edge, he can use the fact that there is only one edge between the group of t and the group of v to identify t and v with nodes in the anonymized graph. Likewise, learning (u, w) allows u and w to be identified with the nodes that represent them. As a consequence, the attacker can infer that (u, v) is an edge, no matter how many other nodes are in the groups.* □

This example shows that revealing an edge can potentially allow an attacker to learn more about the nodes that it connects, and thus infer more about the connections between such nodes. However, the amount revealed about entities for which the attacker does *not* have information is minimal. A relaxed definition of grouping allowing groups of size one enables this intuition to be formalized.

Definition 4.4.4. *Define a $(k, \ell)^{*(p,q)}$-grouping as a grouping in which removing at most p nodes from V leaves a k-grouping of V, and removing at most q nodes from W leaves an ℓ-grouping of W.*

Observe that a (k, ℓ)-grouping is also a $(k, \ell)^{*(0,0)}$-grouping. Also, by applying Lemma 4.4.1, we note that a safe $(k, \ell)^{*(p,q)}$-grouping still gives a lot of privacy for nodes in the grouping: between a group of size k and one of size ℓ, each possible edge is present in at most a $1/\max(k, \ell)$ fraction of possible configurations, as before. But also, between a group of size 1 and one of size ℓ, there can be at most one edge in a safe grouping, and (also by Lemma 4.4.1) the edge is present in at most a $1/\ell$ fraction of possible configurations. Symmetrically, between a group of size k and one of size 1, the (at most one) edge is present in at most a $1/k$ fraction of possible configurations. Only between two groups of size one can we infer the existence (or absence) of an edge with certainty.

Theorem 7. *In the learned link case, given a safe (k, ℓ)-grouped graph and $r < \min(k, \ell)$ true edges, the most an attacker can infer corresponds to a $(k - r, \ell - r)^{*(r,r)}$-grouped graph.*

Proof. This is shown by induction over the revelation of r edges. The base case $r = 0$ yields the $(k, \ell)^{*(0,0)}$-grouped graph. In the inductive case, there is a $(k - r, \ell - r)^{*(r,r)}$-grouped graph, and an additional edge (v, w) is learnt. As shown in the example above, in the worst case, this is enough to identify which node in the anonymized graph is v and which is w. This corresponds to refining of the groups: if v was in a group of size at least $k - r$, it is effectively split into a group of size 1 (containing v alone), and the remaining nodes now form a group of size at least $k-r-1$. Likewise, the group containing w is split into one of size 1 containing w alone, and one of size at least $\ell-r-1$. The resulting grouping is therefore at least a $(k - r - 1, \ell - r - 1)^{*(r+1,r+1)}$-grouping.

Observe however, that the identification of v and w reveals nothing about any other nodes, even those connected to v and w. More precisely, the resulting grouping is still *safe* by Definition 4.4.3. The crucial observation is that any refinement of a safe grouping by partitioning groups into smaller pieces remains safe. By appealing to Lemma 4.4.1, the attacker cannot infer any associations beyond those that are revealed by the grouping directly (i.e. only those links between groups of size one). This is sufficient to bound the new knowledge by the $(k - r, \ell - r)^{*(r,r)}$-grouping. □

This is directly comparable to results on tabular data k-anonymization where the aim is to ensure that individuals are secure up to the revelation of $k - 1$ pieces of information about other individuals. Here, individuals and their associations are secure up to the revelation of $k - 1$ pieces of information (edges) about others.

4.4.4 Query answering on (k, ℓ)-grouped graph

We show that aggregate queries of the type considered in Section 4.2.3 can be answered accurately and efficiently from a published (k, ℓ)-grouped graph. First, since E' is isomorphic to E and queries of type 0 are solely on the underlying graph structure, they can be answered exactly. Queries of type 1 and 2 cannot guarantee perfect accuracy, since it is not possible to to determine exactly which nodes their predicates select. However, they can be answered approximately, by providing bounds and expected values on the aggregate query. It is beyond the scope of this paper to cover all possible such queries, so we instead analyze various typical cases.

A typical type 2 query is of the form "count the total number of papers by US authors in SIGMOD", Since the papers within each group is known, the number of nodes selected by the paper predicate in a group is easily found. The same is true for any author group. The tightest bounds

follow from evaluating the query over all possible assignments of entities to nodes, but this would be very costly, as the following theorem argues.

Theorem 8. *Finding the best upper and lower bounds for answering an aggregate query of type 2 is NP-Hard.*

Proof. The hardness of the tight upper bound problem is shown by a reduction from the set covering problem [27]. Given subsets $S_1, \ldots, S_t$, whose union is $U = \{a_1, \ldots, a_u\}$, construct a bipartite graph (V, W, E). For each subset S_i, create a node v_i in V. All nodes in V are placed into a single group of size t. For each $a_i \in S_j$, create a node w_{ij} in W, and an edge (v_j, w_{ij}). W is partitioned into groups corresponding to the same a_i, i.e., group $G_i = \cup_j \{w_{i,j}\}$. The grouping of the graph is safe, by construction. To decide whether there exists k subsets that cover U, we set our problem as follows: the query selects k nodes in V, and exactly one node from each group of W. There is a set cover of size k if and only if the answer to the tight upper bound problem is $|U|$.

The hardness of the tight lower bound problem is shown by a reduction from the maximum independent set problem [27]. Given an undirected graph $G_1 = (V_1, E_1)$, construct a bipartite graph $G' = (V, W, E')$ similarly to the proof of Theorem 6: for each edge $(v_i, v_j) \in E_1$, insert $(v_i, (v_i, v_j))$ and $(v_j, (v_j, v_i))$ into E', and create a group of size 2 containing the two nodes (v_j, v_i) and (v_i, v_j). All nodes in V are put in a single group. Again, the grouping of G' is safe by construction. To decide if there exists an independent set of size k in G, set the query to select k nodes in V, and only one node in each group of W. There is an independent set of size k if and only if the tight lower bound for this query is 0. □

Instead, slightly weaker bounds are obtained by considering each pair of groups in turn to find bounds on the query answer:

Example 7. *Consider answering the query "Count the total number of papers by US authors in SIGMOD". Given a safe group AG_i of k_i authors, of whom a_i are US Authors; a safe group PG_j of ℓ_j papers, of which b_j are SIGMOD papers; and c_{ij} edges between the two groups: (i) There can be a contribution of at most $U_{i,j} = \min(a_i, b_j, c_{ij})$ to the query (Upper Bound). The total contribution over all author groups AG_i is at most $U_j = \min(\sum_i U_{i,j}, b_j)$, and the final bound over all paper groups is $U = \sum_j U_j$. (ii) There is a contribution of at least $L_{i,j} = \max(0, a_i + b_j + c_{ij} - k_i - \ell_j)$ to the query (Lower Bound), and the bound over all author groups is $L_j = \max_i L_{i,j}$. We can sum this to get the overall lower bound, $L = \sum_i L_i$. (iii) Treating all assignments of nodes to entities as equally likely, the expected selectivity between AG_i and PG_i is $E_{ij} = \frac{a_i b_j c_{ij}}{k_i \ell_j^2}$ (Expected*

Bound). Over all groups, the estimated expected bound is $E_j = \ell_j(1 - \prod_i(1 - E_{i,j}))$*, assuming independence and using inclusion-exclusion principle. The expected bound for the query is then* $E = \sum_j E_j$*. These can be verified by simple case analysis over the structure in Figure 4.5(a).* □

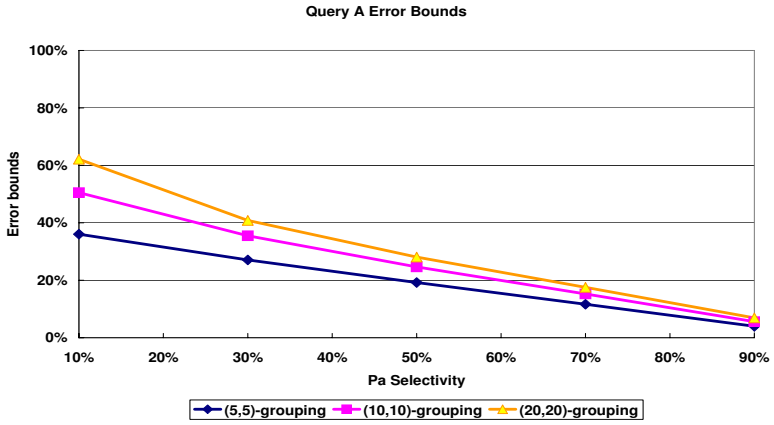
Such queries can be answered in time $O(|E|)$, since each edge in the original graph connects a single pair of groups, and for groups with no edges between them ($c_{ij} = 0$), $U_{i,j} = L_{i,j} = E_{i,j} = 0$.

Example 8. *The query "Find the maximum number of single author papers in VLDB by a US author" can be answered by considering in turn each node that could possibly be a US author (is in a group which contains at least* $a_i \geq 1$ *US authors), and finding exactly the single-author papers associated with that node. Upper and lower bounds increase by one if there are* $b_j \geq 1$ *VLDB paper or fewer than* ℓ_j *VLDB papers in the paper's group of size* ℓ_j*, respectively. These imply upper and lower bounds on the global maximum. Similarly, expected bounds follow by assuming an author has probability of being a US authors with probability* a_i/k_i *in a group of* k_i *authors; and that a paper in a group of* ℓ_j *papers has probability* b_j/ℓ_j *of being a VLDB paper.* □

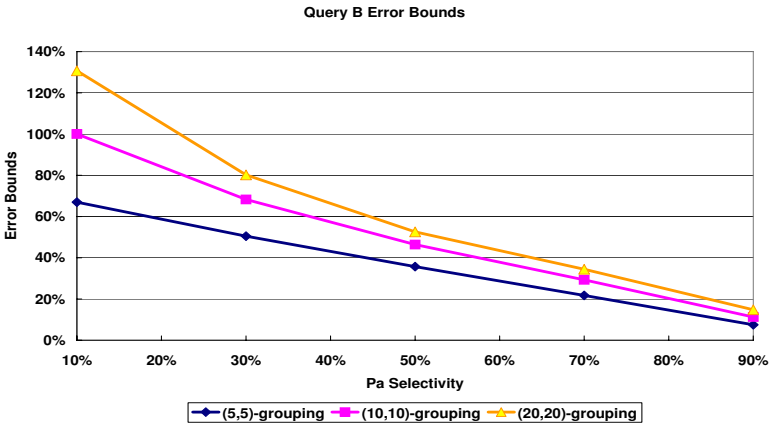
As above, since we have to do a constant amount of work for each edge in the original bigraph, the computational cost is $O(|E|)$.

4.4.5 Finding a safe grouping

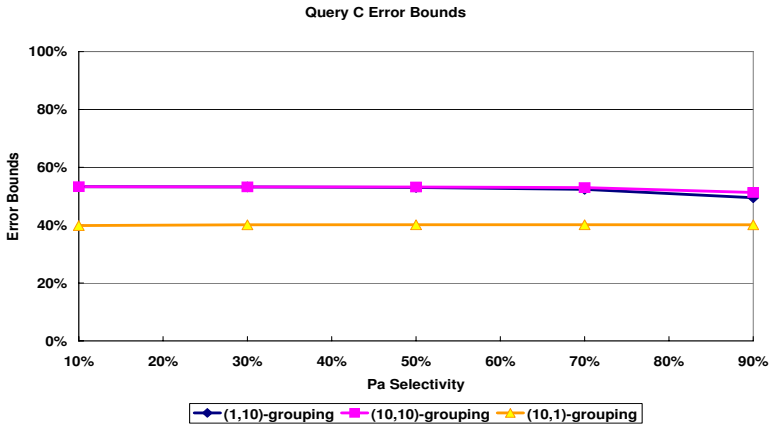
We describe a greedy algorithm which attempts to find a safe k-grouping. Precomputing the self-join of the edge table E on W allows quickly testing whether it is safe to put two nodes in the same group. For each node v in turn, the algorithm attempts to place v in the first group of the partial grouping with fewer than k nodes. If this would make the grouping unsafe, it tries the next group, and so on. If there is no group that meets these requirements, then a new group is started, containing v alone. After processing all nodes, there may be some (few) groups number with fewer than k nodes in them. The algorithm collects these nodes together, and reruns the above loop allowing for groups of size $k + 1$ instead of k. If the graph is sufficiently sparse, then a safe grouping in which every group has either k or $k + 1$ nodes in is produced, and so the grouping is strict. Else, the algorithm continues but now allows groups up to size $k + 2$, and so on. Eventually, either a safe grouping is found, or the algorithm terminates once some large group size is reached. In this case, the method fails, but can be run again by choosing a different ordering of the nodes, or by picking a smaller value of k. Pseudo-code of this heuristic is shown in Figure 5.



(a) Query A



(b) Query B



(c) Query C, P'_a selectivity 0.8

Figure 4.6: Impact of query selectivity and group size on three queries

In our experiments it was easy to find safe k-groupings for small values of k. There is the opportunity to optimize by choosing an initial ordering for the nodes, with the aim of giving better accuracy on queries. When a selective predicate is evaluated over a group, tighter query bounds are given when either (almost) all nodes in the group are selected, or none are selected. When a handful of nodes are selected from a group, there will be more uncertainty in answering the query. Putting similar nodes in a group together will therefore give higher accuracy. It is tempting to do this based on attributes of the entities. However, this can lead to attacks in the style of the minimality attack defined in [18]: knowing that groups were formed in a particular way allows an attacker to deduce the identity of nodes, and hence infer associations. Instead, if groups are chosen solely on graph properties, then the attacker will have no additional information, as the grouping is formed using only open information (recall, the underlying graph is public).

This still gives many possibilities. For example, if to improve accuracy on queries involving graph properties such as node degree (e.g. selecting single author papers), then sorting by node degree will greatly improve query answering. The sorted list of degrees of neighbors can break ties. Other arrangements are possible, but in our experimental evaluation we will compare the groupings found by an arbitrary ordering of the nodes to one based on first sorting in the manner we have outlined.

4.4.6 $(k, 1)$- and $(1, \ell)$-Groupings

A significant class of groupings arise when all groups of one set of nodes are of size 1. These are $(k, 1)$- (or symmetrically, $(1, \ell)$-) groupings. Here, more is revealed about associations between entities of the *same* type (our focus up to now has been on associations between entities of *differing* types), since the true mapping from one set of nodes to entities is revealed. In the author-papers example, a $(1, \ell)$-grouping reveals exactly how many papers a particular author has written, who their coauthors are, which sets of authors have collaborated together to write papers, and so on.

Example 9. *Figure 4.5(c) shows a safe $(1, 3)$-grouping of our example data. The corresponding published tables are H_W and R_W as shown in Figure 4.3(b); H_V and R_V are not needed, since V maps directly onto the nodes of E'. Despite this information being revealed, the private associations between authors and papers are still hidden: although Figure 4.5(c) shows that authors a1 and a3 collaborated on a paper, it could be any one of $\{p2, p3, p5\}$.* □

This again resembles ℓ-diversity: any author is known to have written one paper out of a group of ℓ. Checking the guarantees of Lemma 4.4.1 and Theorem 7, given a safe $(k, 1)$-grouping,

any edge still is between one of k equally likely nodes of V, and given r edge revelations, an attacker is still faced with a $(k-r,1)^{*(r,n)}$-grouped graph. While information is revealed about interactions between one set of nodes (authors, in the example above), in many cases, this information release may be permissible. Our above results show that there are still strong guarantees on the privacy of associations, while revealing information such as, from pharmacy sales, which medicines were bought by the same person (without revealing who that person is). If it is acceptable to release such information, some queries can be answered with higher accuracy.

Query answering on $(1,\ell)$ and $(k,1)$-grouped data. Queries are answered in much the same way as in the more general (k,ℓ) case. However, many queries are answered more accurately, since the amount of uncertainty is reduced: in Examples 7 and 8, $a_i = k_i = 1$ or $b_j = \ell_j = 1$, simplifying the bounds. In particular, some queries of type 1 can be answered exactly: if the predicate is on the 1-grouping, the correct set of entities can be found exactly, which allows the exact answer to the aggregate query to be found. Type-2 queries can be answered with tighter bounds:

Example 10. *For the query of Example 7 over a $(k,1)$-grouped graph, the set of SIGMOD papers is known precisely. For each of SIGMOD paper, we add 1 to the upper bound if they have an author in a group which contains a US author; and add 1 to the lower bound if they have an author in a group in which every author is in the US. For the expected bound, the probability that an author in a group of size k_i with a_i US authors is $E_{i,j} = a_i/k_i$, so the probability of any author of the paper being from the US is $1 - \prod_i (1 - E_{i,j})$. Similarly, for Example 8, we can consider all single author papers by US authors exactly, and find the corresponding bounds (upper, lower, and expected) on which are in VLDB.* □

4.4.7 Experimental Analysis of Utility

In this section, we evaluate the utility of the anonymized data through experiments on the DBLP data. Specifically, we study the accuracy of three queries with different properties. For each query, we compute the lower bound estimation L, the upper bound estimation U, and the expected value E. If the correct answer to the query is Q, we compute two error measurements: the error bounds $\frac{U-L}{2Q}$ (the worst case error from using $(U+L)/2$ as an estimate for Q), and the expected error $\frac{|E-T|}{T}$. The three queries are:

- **Query A:** *Find the average number of authors of any paper satisfying predicate P_a.*

This is a type-1 query with an attribute predicate only. We vary the selectivity of P_a from 10% to 90%.

- **Query B:** *Find the total number of single author papers satisfying* P_a. This is also a type-1 query with both attribute predicates and structural predicates. The selectivity of P_a is varied as above, while the single author predicate is held constant.

- **Query C:** *Find the total number of papers satisfying* P_a *having authors who satisfy* P'_a. This is a type-2 query. We vary the selectivity of both P_a and P'_a.

These fit exactly the form of the queries we have studied in Example 7 and Example 8 (note that type-1 queries can be thought of as type-2 queries where one of the attribute predicates is always true). We do not consider any type-0 queries, since our earlier analysis shows that they can be answered exactly from the graph structure alone. We computed groupings over the papers and authors in the DBLP data described in Section 4.2 using the method detailed in Section 4.4.5. We built 20-groupings, 10-groupings, and 5-groupings over the data. The first iteration of the algorithm was able to find safe k-groupings covering almost every node: the 20-grouping of papers had 43 papers (out of 540K) not in groups of size 20, while there were just 3 authors not in groups of size 20. The next iteration easily found a safe, strict 20-grouping.

The following parameters can impact query accuracy:

- Group size: We compare approaches from $(k, 1)$, $(1, \ell)$ and (k, ℓ) groupings. We expect smaller group sizes to offer better accuracy for query answering.

- Selectivity of predicates: More highly selective queries are more likely to touch just a few nodes within a single group, and so lead to wider worst case bounds.

- Grouping formation: We will study the impact of building the groupings based on an arbitrary initial ordering of the nodes, and based on sorting based on degree and neighborhood degree, as discussed in Section 4.4.5. We expect sorted groupings to give better answers when queries have structural predicates based on degree.

Worst Case Error Bounds. Figure 4.6 shows the worst case error bounds for query answering with (k, k)-groupings over the queries A, B, and C. As expected, smaller groupings achieve smaller error. There is also a clear trend for Queries A and B (Figure 4.6(a) and 4.6(b)) that as the selectivity of P_a increases, the accuracy improves. When only a single node in a group is touched by a query, as happens when selectivity is low, it could be any node, and so we have high uncertainty for the aggregate value in the group. But when many nodes are selected in a group, there is less relative uncertainty for an aggregate like sum or average.

For Query C (Figure 4.6(c)), there is little variation as we vary the selectivity of P_a (in this plot, selectivity of P'_a is set to 0.8). Note that when we have a paper group of size 1, as in

the $(10, 1)$-grouping, we can directly select out exactly those papers that meet the predicate, and so have better accuracy compared to other groupings. There is little difference between the $(10, 10)$-grouping and the $(1, 10)$-grouping. This is because P'_a selects most authors, so there is not much benefit from the $(1, 10)$-grouping's ability to eliminate some candidates. When P'_a selects fewer authors, there is a clearer advantage of $(1, 10)$ over $(10, 10)$ grouping.

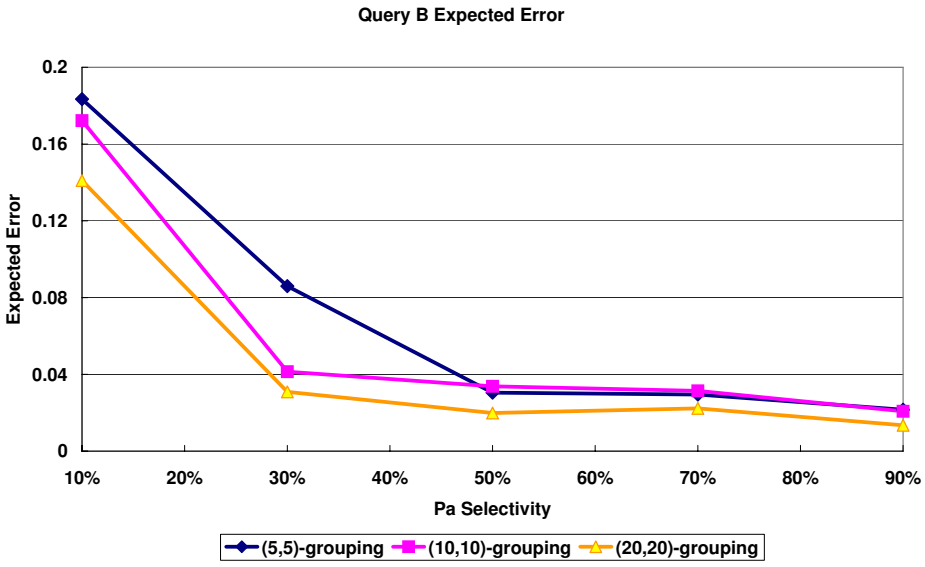
Expected Case Error Bounds. Although the above worst case bounds show that there can be a wide range between the upper and lower bounds on a query, we show next that the expected bound can give a quite accurate answer. Figures 4.7(a) and 4.7(b) show the expected error on queries B and C (the expected error on query A was too close to zero to plot). The general trend is again that higher values of selectivity give better accuracy. On query C, as in the worst case, the expected error is much smaller on $(10, 1)$ than $(1, 10)$ or $(10, 10)$, which are about the same for this (more selective) P'_a.

Impact of ordering on grouping. Query 2 involves a structural predicate (single author papers), so we compare ordering the grouping by degree and second-order degree to an arbitrary grouping in Figure 4.8(a). We see that there is a very dramatic benefit to having a grouping based on this ordering: two orders of magnitude improvement in the accuracy. This is because most groups now contain papers with the same number of authors, meaning the contribution to the aggregate can be found exactly for those groups, and the few remaining contribute to the uncertainty.

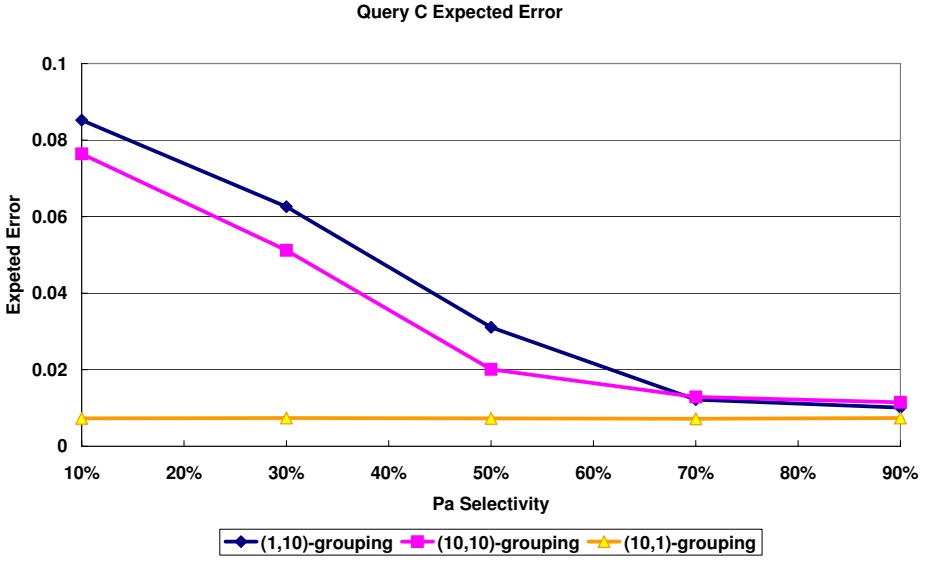
We further investigate the impact of the correlation of the predicate with the grouping. For query B, we construct an artificial predicate P_a which selects the same number of total papers, but touches a variable number of papers in each paper group within a $(5, 5)$ grouping. Figure 4.8(b) shows that as the number of papers touched in each group increases, both the expected and worst case bounds improve up to the point when all papers are selected in a group, the aggregate query is answered with perfect accuracy. This shows that if we can anticipate the kinds of structural predicates that end users will want, then we can improve the utility of the published data without compromising the privacy.

4.5 Unions of groupings

In this section, we consider the impact of publishing multiple groupings of the same graph. This allows a class of queries to be answered with perfect accuracy, but is open to stronger classes of attack based on the graph structure. Recall that publishing the fully-censored (m, n)-grouped

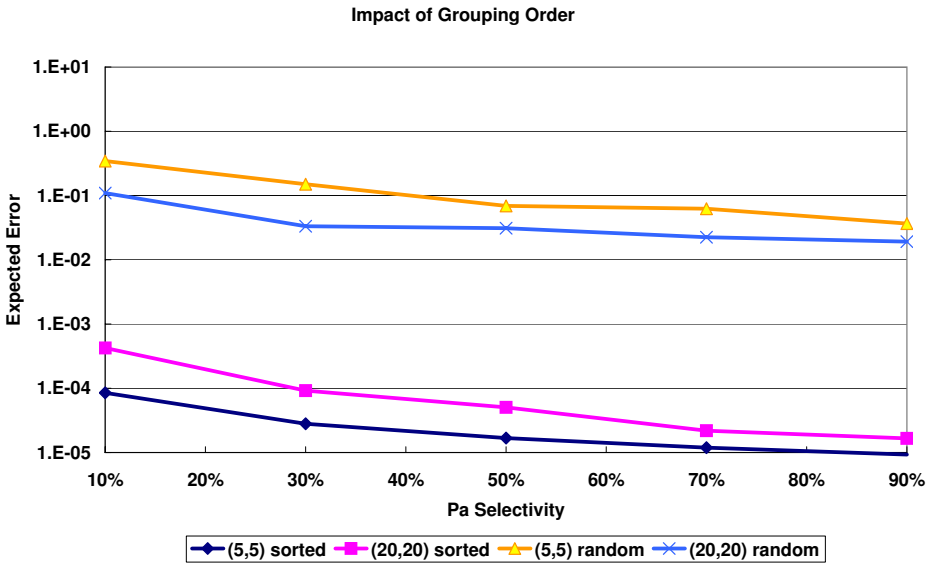


(a) Expected Error on Query B

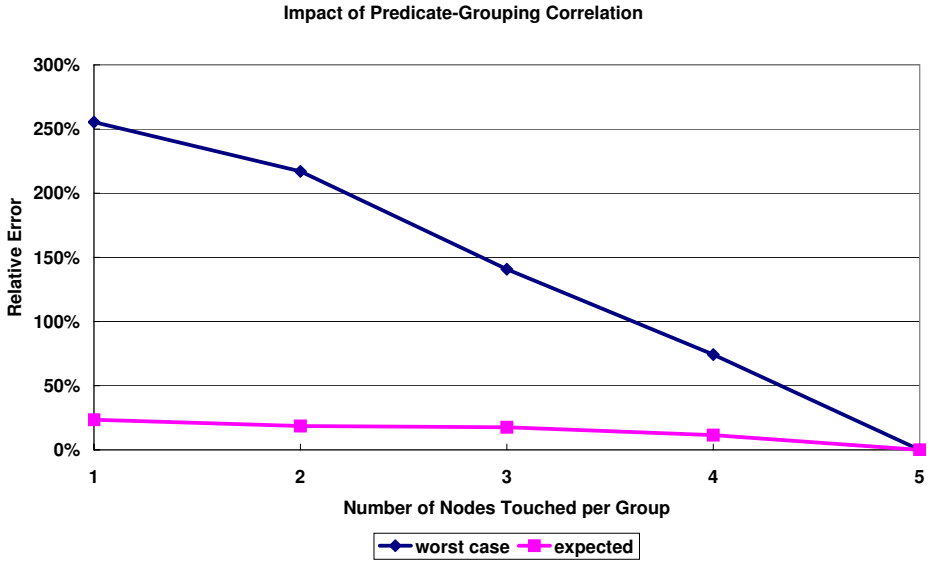


(b) Expected Error Query C, P_a' selectivity 0.1

Figure 4.7: Expected Error Bounds



(a) Impact of Grouping Order



(b) Correlation between predicate and groups

Figure 4.8: Impact of Grouping and Clustering

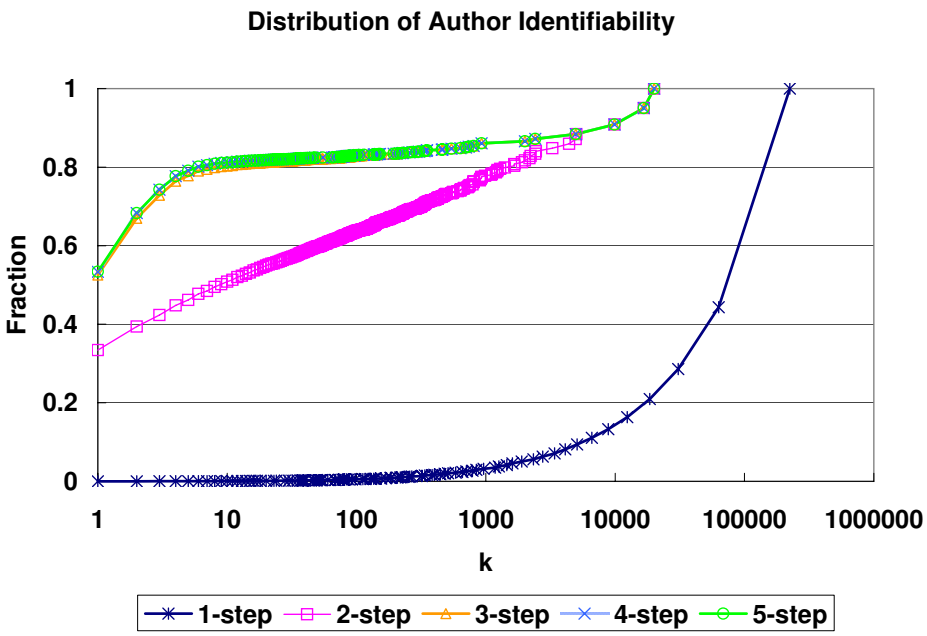


Figure 4.9: Author identifiability with $(m, 1) \cup (1, n)$-grouping

graph allows type-0 queries to be answered exactly, but gives us no handle to answer other query types with any certainty. As observed in Section 4.4.6, publishing $(1, \ell)$ or $(k, 1)$ grouped graphs offers greater utility for a variety of queries while preserving the privacy of associations.

So it seems that publishing both as a $(1, \ell) \cup (k, 1)$ grouping of a graph is desirable, since any query of type 1 could be answered exactly. But while either version of the graph in isolation is resilient against attack, by publishing both an attacker could combine the information in the static attack model (Definition 4.2.1). If an author has written more papers than any other, she can be identified from the $(1, \ell)$ graph. From its unique degree, the same node can be located in the $(k, 1)$-grouped graph, revealing all the papers written by that author. This attack applies even over $(m, 1) \cup (1, n)$-grouping, which give the most privacy in this class.

More strongly, if certain types of mappings between the isomorphic $(m, 1) \cup (1, n)$-grouped graphs can be found, the original data can be recovered. This problem is related to, but distinct from, the (well-studied) graph isomorphism problem. Some information will remain private: associations between authors who have only ever written one single author paper and their papers cannot be recovered from $(k, 1) \cup (1, \ell)$-grouped graphs, even though finding an isomorphism over these nodes is easy. But if the attacker can find node or edge pairs that must be *uniquely* mapped to each other in *every* isomorphism, their identity is compromised. Clearly, the amount of

privacy that remains is input dependent: data consisting solely of single author papers by one-time authors is secure; but if each node has a unique degree then total re-identification is trivial. On realistic data, the truth lies somewhere in between.

Experimental Analysis of Privacy. We define the following attack on publishing $(m, 1) \cup (1, n)$-grouped graphs, based on finding matching pairs of nodes between the two graphs. Each node in the fully censored graph is given a compact *signature*. Initially, the signature of every node is a default value, say 0, since there is no *a priori* way of telling them apart. Given a node, its *next-step signature* is formed by concatenating its current signature with the signatures of all its neighbors in the graph, and sorting this set lexicographically. Once next-step signatures are found for all nodes in the graph, they can be compactly relabelled (since there can be at most n different signatures for n nodes). By this construction: (a) If two nodes have different signatures then they cannot be matched in any isomorphism—since the signature canonically encodes features of the neighborhood of a node, different signatures entail non-isomorphic neighborhoods. (b) Since the process is entirely deterministic, each node will obtain the same signature every time the procedure is run on the graph. As a result, if a node receives a signature that is not shared by any other node, then this node must be uniquely matched in any isomorphism. Moreover, it can be matched to the unique node with the same signature in an isomorphic copy. Note that the implication is only one way: the guarantee is that if signatures are unique then nodes can be uniquely matched, and not vice-versa. Schemes which build signatures for edges instead of for nodes are also feasible; the details are quite similar, and we present results only for the node-based scheme for brevity.

We apply this signature scheme on the $(m, 1) \cup (1, n)$-grouped graphs, and measure how many nodes are uniquely identified, and how many fall into equivalence classes of size 2, 3, 4 etc. The cumulative distribution over such classes of authors in the DBLP dataset are shown in Figure 4.9. Multiple steps of signature computation were performed, but on this data (and on similar datasets such as IMDB), no improvement was seen after the forth iteration, and there is only limited difference from the third to forth step. A 4-step signature is sufficient to identify half of authors uniquely, and only 20% are in equivalence classes of 10 or larger. Such privacy levels are weak for many typical applications, so we conclude that $(1, \ell) \cup (k, 1)$-groupings should be avoided. The single groupings discussed in the previous section offer much stronger privacy guarantees while allowing queries to be answered accurately.

4.6 Summary

We have considered the problem of anonymizing data in the form of bipartite graphs, and shown that methods based on finding safe (k, ℓ)-groupings are effective at securing published data against a variety of attacks. There are several directions for future work:

- We have shown how to answer queries for various natural classes of aggregates, but it remains to automatically rewrite arbitrary queries to give upper, lower and expected bounds on safely grouped graphs. It is also of interest to study advanced query types, such as join-style queries over longer edge paths.

- We have assumed that full information can be revealed about entities, but the mapping from entities to nodes in a graph must be masked. Other models may be needed if we wish to anonymize both entities *and* the associations between them.

- Our focus has been on the important class of data that can be represented as a bipartite graph linking two types of entity. It is natural to also study arbitrary graphs over a single type of entity, i.e. the social network graphs that have so far frustrated attempts at anonymization [4, 26]. A natural approach would be to extend the definition of safe groupings to ensure that a group of k nodes has no edges between nodes in the group, and no common neighbors, and to analyze what privacy guarantees hold over such groupings.

Chapter 5

Personalized Privacy

5.1 Motivation

In chapter 3, we provide flexibility to data owners to specify a target distribution, such that after generalization and permutation on the sensitive attributes, the distribution of any user's data in the table can be inferred only as the target distribution, but nothing else. From the experimental results we observe that in many cases the error rates of query answering are big under this strong privacy requirement. This is because what we have proposed is a uniform privacy guarantee. Each user's privacy is protected at the same level. In order to satisfy such a strong privacy requirement, we have to sacrifice the utility of the anonymized tables.

In practice, people have different privacy requirements. Some people may care about their privacy very much, while some others may not care about it at all. If we can relax the privacy for those users who are not so concerned about their privacy, empirically we can reduce the generalizations involved, and get better query answering. In this chapter, we investigate techniques to improve query answering accuracy while preserving each individual's personal privacy requirement. This personalized privacy model is comparable with the work in [8]. However, their model only works with categorical data, which is similar to previous works such as k-anonymity and l-diversity. And it is also based on the QI-generalization mechanism. The privacy model we proposed is an extension to the distributional privacy we proposed in chapter 3. It not only addresses the privacy requirements of numeric data, provides more flexible privacy specifications to data owners, but also has much better query answering accuracy.

In our proposed model, each user i having a tuple in the table will be asked to specify a node H_i in the privacy-annotated hierarchy H as his/her privacy preference R_i. After anonymiza-

Table 5.1: An example microdata table

	ID	Quasi-identifiers		Sensitive	
tuple ID	name	zipcode	gender	salary	Privacy Requirment
4	Debra	91130	F	\$60K	[\$60K]
5	Elaine	91210	F	\$40K	[\$30K, \$40K]
6	Grace	91220	F	\$30K	[\$30K]

tion, the published data should respect the privacy requirement of all R_is. We will also identify the optimization problem to provide accuracy query answering, and design efficient algorithms to achieve our goal.

5.2 Example

In this section, we use some examples to illustrate the proposed privacy goal and the basic idea of the new anonymization scheme. We also show that by relaxing the privacy guarantee for those users whose privacy requirement is not so strong, we are able to provide more accurate query answers compared with the distributional privacy.

We use the same example microdata table as shown in table 3.1, and the annotated hierarchy is the same as that in figure 3.1. Each user will choose a node in the hierarchy tree as the specification of his privacy requirement. If a user choose a node with domain D, it means the user's sensitive attribute value is in the range of D, and after publishing the anonymized table, for all values in D, their relative probability should follow the target distribution P. For example, suppose a user chooses the node with domain $[\$50K, \$60K]$ from the tree in figure 3.1. Then in the anonymized table, anyone should only infer that his salary is either $50K$ or $60K$ is equally likely. With this definition, the generalized and permuted domain generalizations $\{50K, 60K\}$, or $\{30K, 40K, \{50K - 60K\}\}$ both satisfies his privacy requirement.

In the microdata table, suppose we have collected all the users' personal privacy requirements, based on the hierarchy in figure 3.1. We will add an additional column to represent the privacy requirement chosen by each user. The resulted table is shown as table 5.1. For simplicity, we only show the tuples from group 2.

One possible way to do SA generalization permutation is shown in table 5.2. By comparing the output with the generalization result of group 2 in table 3.2, we can easily see the range of generalized domains are much smaller than the previous scheme, and we can further split them into 3 groups. This is because the range of their privacy requirements are disjoint with each other. The

Table 5.2: The example table after sensitive attribute generalization and permutation

		Quasi-identifiers		Sensitive
group ID	tuple ID	zipcode	gender	salary
2a	4	91130	F	\$60K
2b	5	91210	F	[\$30K, \$40K]
2c	6	91220	F	[\$30K]

resulted generalized domains after we do grouping and permutation within each group satisfies the privacy requirements specified by each user.

Given a microdata table, there are many ways to partition it into multiple groups and perform sensitive attribute permutation and generalization. In the remaining part of this chapter, we will formally define the personalized privacy problem, and then identify proper optimization goals that can improve the accuracy of query answers.

5.3 Personalized Privacy Model

After the data publisher specifies a hierarchy tree H, each user i with sensitive attribute value v_i will chose a level h_i as his/her privacy preference. Denote the leaf node in the tree corresponding to v_i as n_i. The node corresponding to the privacy requirement that v_i specifies is the node in the path from n_i to the root, and whose height is h_i. Denote the domain of this node as R_i, obviously $v_i \in R_i$. We also refer R_i as the privacy specification of user i. Now given a multiset G of subdomains of the the sensitive attribute domain $\mathcal{D}(S)$, we can compute the probability of any element $x = t$ as $Prob(x = t|G)$, after we permute all the domains in G.

Definition 5.3.1. *Let $G = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a multiset of subdomains of $\mathcal{D}(S)$, and R_i be the privacy preference specification of user i. We say G* respects *R_i if for any elements $t_1, t_2 \in R_i$, if $P(x = t_2) \neq 0$, then $\frac{P(x=t_1|G)}{P(x=t_2|G)} = 1$.*

This definition says, for all the values in the range R_i specified by user i, their probability should follow a uniform distribution in G after permutation of all the generalized domains. For example, $G = \{[1-2], [1-2], 3, 4\}$ respects $R = [3-4]$, as each value $1, 2, 3, 4$ has a probability of $1/4$ after domains in G are randomly permuted.

Suppose we have N tuples in the database, and we will collect N privacy preference $R = \{R_1, R_2, \cdots, R_N\}$. We say a multiset of subdomains G respects R if it respects R_i for all $1 \leq i \leq N$.

Definition 5.3.2. *Let $S = \{s_1, \ldots, s_n\}$ be a multiset of elements in $\mathcal{D}(S)$, $G = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be a multiset of subdomains of $\mathcal{D}(S)$. We say G is a* R-respect generalization *of S, if G respects each R_i, and there is a one-to-one mapping f from S to G, such that for all s_i, $i = 1, \ldots, n$, $s_i \in f(s_i)$.*

Intuitively, if an attacker only has access to a R-respect generalization, then for each user i's sensitive attribute, within the range of R_i he can not derive more information of the values in R_i other than a uniform distribution.

Once we get a R-respect generalization, answers to the aggregate queries can be obtained by first pre-computing the help table, which contains the bounds of the query answers when any number of tuples in a partition is touched. Then for any query we only need to identify how many tuples in each partition is touched, then join with the help table to get the bounds and aggregated them.

A trivial R-respect generalization is to produce the same generalization as we've proposed in distributional privacy. This can be seen as setting all the R_is to the whole domain. In the personalized privacy model, the privacy requirement is relaxed. We can expect that in many cases we do have better choices of the generalized domains which lead to more accurate query answers. In the next section, we will formally define the optimization problem. We then present algorithms that can produce generalizations which conforms to the personalized privacy requirement, and meanwhile, can answer aggregate queries accurately.

5.4 Optimization for Accurate Query Answering

We have already identified the personalized privacy goal, and given the formal definition based on the hierarchy framework. Next we will design optimization criteria for anonymization in order to provide accurate query answers.

Intuitively, the lower a subdomain D is in the domain hierarchy, the narrower range $|D|$ it has, and will introduce less uncertainty in query answers. So our first optimization goal is to minimize the range of generalizations. We formally define the problem as following:

5.4.1 Problem definition

Problem 5.4.1 (Optimal Personalized Privacy Problem). *Given a multiset S of sensitive attribute values, a domain generalization hierarchy H over $\mathcal{D}(S)$, and the privacy specification R of all users, find a R-respect generalization G of S such that $\sum_{\mathcal{D}' \in G} |\mathcal{D}'|$ is minimized.*

For short, we also call G the optimal R-respect generalization of S, and the personalized privacy problem as PPP.

Case 1:

We don't generate any subgroups for each group. That is, we put the generalizations R_i of all v_i in one group and permute them.

Lemma 5.4.1. *If there's one tuple with (value, privacy) pair (v_1, R_1), and another with (v_2, R_2), and satisfies $R_2 \subseteq R1$, then we can raise v_2's privacy to R_1.*

Proof. Denote the generalization group as G. After permutation, G must satisfy that all values in the range of R_1 and R_2 follow uniform distribution. Because $R_2 \subseteq R1$, so R_1 follows uniform distribution automatically make true that R_2 also follows uniform distribution. This indicates that although v_2's privacy is set as R_2, the output is the same as it is set as R_1. So we can replace R_1 as v_2's privacy requirement. □

With Lemma 5.4.1, we can derive the following theorem.

Theorem 9. *For any (value, privacy) pair (v_i, R_i), its privacy level can be raised to $max(R_j)$, where $v_i \in R_j$.*

□

And we have the following algorithm 5.4.1 to solve for the optimal solution of this special case.

Algorithm 6 MinGenUniformOneGrp(T, P)

```
//input: a hierarchy tree T, a set of (values, privacy) pair P_i = (v_i, h_i).
//output: a group of generalized domains G.
Put each pair P_i at its privacy specification node h_i;
level = root;
BigDomains = GetBigDomains(level); //this function returns the biggest disjoint domains that
cover the whole tree. Details omitted here.
for each domain D_i in BigDomains do
    V=all values in the subtree rooted at D_i;
    generalize(V); // this is the generalization algorithm we already have.
end for
```

Note that in this case, the same values within the same group will have the same privacy requirement, as for any $(v, R_1), (v, R_2), \cdots$, their privacy will be promoted to the biggest R_i according to Lemma 5.4.1.

Another observation is that, when the BigDomains is not the root, we can further separate all the values V into |BigDomains| subgroups, each of them contains all values from the subtree rooted at each element of BigDomains, and generalization are within each subtree respectively. It is obvious as the subtrees are disjoint with each other.

Case 2:

In general we may want to have multiple groups. It will enhance the accuracy of query answering. One possible solution to this case is that, we design a scheme to partition the data into multiple groups, then for each group apply the solution of case 1. One such heuristics is to group tuples by their quasi identifiers, as shown in algorithm 5.4.1.

Algorithm 7 MinGenUniformMultiGrp(T, P)

```
//input: a hierarchy tree T, a set of (QI, SA, privacy) triple P_i = (q_i, v_i, h_i).
//output: groups of generalized domains G.
Group tuples based on q_i;
for each group do
   call MinGenUniformOneGrp;
end for
```

Although grouping based on quasi identifiers can increase the number of tuples touched in each partition, it doesn't taken into consideration the properties of sensitive attribute values. As demonstrated in previous work, optimizing the overall sum of generalized ranges will give the best query accuracy.

A heuristic algorithm to solve the PPP problem works as following:

1. Given a balance binary tree on the domain, we assign all values to each leaf node.

2. We first extract the partitions which contains all leaf nodes, until some leaf node doesn't have any value left.

3. Prune the tree, and it's now unbalanced. The "leaf" level now contains nodes which are previously inner nodes of the tree.

4. For each node in the new "leaf" level, find the partition which gives the maximum saving. For any value v_i at that node, the saving is computed as $s_i = |R_i| - |D_i|$, in which R_i is v_i's privacy

requirement, and D_i is the generalized domain. The maximum saving for a partition g is $\sum_{\forall s_i \in g} s_i$.
5. For all nodes in the leaf level, extract the group with the maximum saving.
6. Prune the tree again and continue step 4.

However, we've find a counter example to show that step 2 is not optimal already: suppose we have a set of values with their privacy requirements (1, 1-8)x4, (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8), where $(a, b) \times c$ means we have c tuples whose values are a, and its corresponding privacy requirement is b. The best partition and generalization should be $g_1 = \{1-4, 1-4, 1-4, 1-4, 5, 6, 7, 8\}, \{2\}, \{3\}, \{4\}$ That is, we group the 4 tuples of 1 together with 5,6,7,8. But step 2 we will output $g_2 = \{1, 2, 3, 4, 5, 6, 7, 8\}, \{1-8\} \times 3$.

We notice the maximum saving is still valid in this case, g_1 has a bigger saving than g_2. So actually we can remove step 2 and start from step 4 directly. However, we do find another counter example that the maximum saving also cannot give the optimal solution: Suppose we have (1, 1-16), (5,5-8), (6,5-8),(7,5-8),(8,5-8),(9,9-12),(10,9-12),(11,9-12),(12,9-12),(16, 1-16) The partition with the maximum saving is 1-4, 5-8, 9-12, 13-16. Saving is 12+0+0+12 = 24. And the resulted partition would be $\{1-4, 5-8, 9-12, 13-16\}, \{5-6, 7-8\}, \{5-8\}, \{9-10, 11-12\}, \{9-12\}$. Savings for the 5 groups are 24, 4,0,4,0 respectively. However, the optimal solution should be: $\{1-8, 9-16\}, \{5, 6, 7, 8\}, \{9, 10, 11, 12\}$. Savings for the 3 groups are 16, 12, 12 respectively.

The modified algorithm is to compute the average saving of each group. This solves the above problem, as the average savings for $\{1-4, 5-8, 9-12, 13-16\}$ is now 24/4=6, while for $\{1-8, 9-16\}$ is 16/2=8. Formally, suppose the domain of v_i's privacy requirement is R_i, and v_i is finally generalized to D_i, then the saving $s_i = |R_i| - |D_i|$. It's possible $s_i < 0$. For each group g, the total saving is $S_g = \sum_{v_i \in g} s_i$, and the average saving is $A_g = S_g/|g|$.

Given a hierarchy tree T, a set of N values (v_i, h_i), where v_i is the SA value of user i, and h_i is user i's privacy specification level in the tree. $h_i = 0$ is the leaf level, the structure of the algorithm is shown as algorithm 8.

But there's still unsolved problem when there are multiple choices in a subtree: Suppose we have (1, 1-8), (3, 3-4), (4, 3-4)x2, (5,5-6), (8, 7-8) The biggest saving is $\{1-2, 3-4, 5-6, 7-8\}$, in which obviously we should generalize 4 to 3-4. But if we select 3 into this group and generalize to 3-4, we will have the same saving. One possible solution when choosing between multiple equivalent values, is to make the left values in the subtree as balanced as possible.

Whether this solution is optimal, is not obvious yet. We need to identify more properties of the problem.

Another possible solution is use a top-down approach as shown in algorithm 9.

Algorithm 8 MinSORbu(T, P): Bottom-up approach to find the optimal groupings and generalizations

//input: a hierarchy tree T, where each leaf node has a set of (values, privacy) pair $P_i = (v_i, h_i)$.

//output: a grouping $\mathcal{G}$.

L = set of all leaves of T;

while there exists value pairs left in T **do**

  for each node n_i in L **do**

    (g_i, s_i)=FindMinGroup(n_i, T, L);

  end for

  extract g as the g_i with the maximum average s_i;

  save g to G

  Prune(T);

end while

Algorithm 9 MinSORtd(T, P): Top-down approach to find the optimal groupings and generalizations

//input: a hierarchy tree T, where each leaf node has a set of (values, privacy) pair $P_i = (v_i, h_i)$.

//output: a grouping $\mathcal{G}$.

while there exists values whose privacy at the root **do**

  output a group of domains using the global generalization algorithm;

  output the tuple ids group of the corresponding generalized values as those with largest R_is;

end while

//now no privacy requirement at root

repeat the above algorithm for each subtree

Unfortunately, our previous example (1, 1-8)x4, (2,2), (3,3), (4,4), (5,5), (6,6), (7,7), (8,8) again serves as a counter example here: the global algorithm will output the generalized domains 1,2,3,4,5,6,7,8 and 1-8x3.

5.5 Summary

In this chapter, we propose a new privacy goal– the personalized privacy. We formally defined it based on the same framework of sensitive attribute generalization and permutation as the distributional privacy in chapter 3. We show properties and hardness of the problem through examples. We also propose heuristical solutions to this problem. As the next step, we will try to either prove the hardness of the problem formally, or identify if there exist polynomial time solutions.

Chapter 6

Discussion

6.1 Summary of contributions

The demand of cross-domain data sharing has been increasing substantially in recent years, when more and more companies are collaborating and outsourcing their data with others. The release of microdata offers advantages for ad hoc analysis. Meanwhile, it also raises privacy concerns when individual records are released.

In this dissertation, I identify the privacy requirements for microdata sharing in several specific scenarios, and propose novel anonymization schemes based on the permutation schemes. The anonymization schemes are optimized to maximize the utility, which is evaluated in terms of query error rates. Specifically, our contributions are as the following:

- ***New anonymization scheme*** Existing anonymization works are mostly based on domain generalization on QI attributes. After generalization, tuples with the same QI values will be put into the same partition, thus it breaks the association between identifiers and QI values. We show that the query error rates resulted from domain generalization are generally too big, and make the anonymized dataset useless. Thus we propose a novel anonymization scheme [22]. We group tuples into partitions and permute the sensitive attributes (SA) within each partition. Privacy is thus maintained by breaking the QI-SA association. We show that given the same partition, permutation-based approach is always better than the generalization-based approach. To improve the query efficiency, we also propose the construction of a help table which pre-computes the bounds when different number of tuples are selected in each partition.

  As pointed out in [18], anonymization schemes based on QI-generalization suffer from min-

imality attack. In our scheme, tuples in the same partition are not based on optimal QI-generalization criteria. So one advantage of our scheme is that, it is resistant to the minimality attack.

- ***Protecting numeric data*** We identified that existing privacy goals such as k-anonymity or l-diversity are often not sufficient to protect numerical attributes. We propose a dedicated privacy definition for numeric data called (k, e) anonymity [22]. In addition to require that each partition has at least k different values, we further require that the range of the possible SA values in a group is no less than e. When anonymizing the microdata, we partition tuples such that each partition satisfies (k, e) anonymity. We also propose several criteria to optimize partitions for accurate answering of aggregate queries.

- ***Disguising data distribution*** In many scenarios, data owners may not want others to infer the actual distribution of portions of the published dataset. We propose an anonymization scheme that ensures an attacker can only derive from the microdata that an individual record's sensitive attribute follows a pre-specified target distribution [28]. Specifically, we build a hierarchy tree on the sensitive attribute domain. Weights are given to nodes in the tree to model the target distribution. For all tuples in the same group, SA values are first generalized along the hierarchy, then permuted within the group. SA values of each tuple can only be derived as following the target distribution then. We identify the optimal algorithm that minimizes the sum of ranges of all generalized domains, while maintaining the privacy requirement.

  When the difference between the target distribution and the source distribution is big, the query accuracy is often poor. We further propose two approaches to achieve better accuracy for query answering. The first approach allows data owners to approximate the target distribution in a controlled manner to tradeoff distributional privacy protection and accurate query answering during anonymization. The second approach is to inject fake tuples into the anonymized dataset.

- ***Anonymizing bipartite graph data*** Graph data pose more privacy challenges than the tabular data. We are the first to propose privacy models for bipartite graph data, and design and optimize the anonymization scheme to accurately answer queries of several well-defined types [29]. Specifically, I first show that existing anonymization techniques for tabular data cannot give accurate answers to queries on graph properties: small changes on edges will change aggregate graph properties radically. We propose novel anonymization schemes for bipartite

graph data, called (k, l)-groupings. We also demonstrate attack with background information of the graph structure, and propose the criteria for a safe grouping. We then study the security properties of (k, l)-groupings in detail. Generally finding a safe grouping is hard, so we also propose a greedy solution. We further classify queries on graph data based on their predicates on the graph structure, and show the query answering process for some typical types of queries, as well as analyzing their hardness.

- ***Personalized Privacy*** We propose personalized privacy model, so that users can specify their own privacy requirements. We give the formal definition of the problem, and show properties and hardness of the problem through examples. We also propose heuristical solutions to this problem. In the future, we will try to either prove the hardness of the problem formally, or identify if there exist polynomial time solutions.

6.2 Related Work

Privacy has raised many concerns in recent years, when data is shared between different parties through Internet. On the one hand, privacy practice needs to be declared when data is collected from individuals. One the other hand, there is great need of mechanisms to ensure that the collected data will be protected from disclosure when later disseminated. Privacy problems have been identified and studied in a broad area of applications [30, 31, 32, 33, 34, 35, 11, 36, 37, 38, 12, 5, 20, 3, 7, 8, 10, 39, 40, 6, 21, 22]. In this section, we only discuss the following works related to our research. First, we introduce the policy-based privacy enforcement, which helps server and client negotiate their privacy practice and requirement. Privacy-preserving data mining is another hot topic these days, when several parties want to collaborate and compute some functions of their data together, or transfer their data to some specialized data miners, which will be discussed next. Our proposed approaches fall in the third part, where the collected data is to be published.

6.2.1 Policy-based Privacy Enforcement

When users share their data with websites, they prefer that the website will not abuse the information they submitted. Policies have been designed to help users and websites to negotiate their privacy practice.

The Platform for Privacy Preferences(P3P) [30] is a protocol allowing websites to declare their intended use of information they collect about browsing users. It enables websites to encode

their data-collection and data-use practices in a machine-readable XML format, known as P3P policies [41]. The W3C has also designed APPEL (A P3P Preference Exchange Language) [42], which allows users to specify their privacy preferences. By adopting these policies a user's agent will be able to check a website's privacy policy against the user's privacy preferences, and automatically determine when the user's private information can be disclosed.

In order for enterprises to effectively enforce their privacy policies in addition to simply specifying them, IBM proposed the Enterprise Privacy Authorization Language (EPAL) [31] as a formal language that provides enterprises with a way to automate and enforce privacy policies. EPAL policies, unlike P3P policies, are enforceable, as they are written and structured in a similar fashion to access control policies that one may find in the security domain. The policies are enforced by an enforcement engine that parses the files, assuring the information collection, use and storage that occurs within the organization, and amongst the organization and its partners, complies with the EPAL specified privacy practices.

6.2.2 Privacy-Preserving Data Mining

Data mining techniques are used to find patterns in large databases of information. But sometimes these patterns can reveal sensitive information about the data holder or individuals whose information are the subject of the patterns. The notion of privacy-preserving data mining is to identify and disallow such revelations as evident in the kinds of patterns learned using traditional data mining techniques.

In some cases several individuals may want to collaborate and evaluate some function of their inputs, such that no more is revealed to a party or a set of parties about other parties' inputs and outputs, except what is implied by their own inputs and outputs. This problem is formally referred to as secure multi-party computation. It was first investigated by Yao [32], and later generalized to multiparty computation. The seminal paper by Goldreich proves existence of a secure solution for any functionality [43].

Sometimes a user do not have the ability to do data mining, and will transfer his data to specialized data miners for analysis. Certain degree of anonymization is needed in order to protect the individual reports or his data. In [12], Agrawal et. al identify the primary task in data mining as the development of models about aggregated data(sum, count, average, maximum, minimum, pth percentile, etc.) without access to precise information in individual data records. They classify the solutions to modify a value in a field into three methods. (1)Value-Class Membership: partition

values into sets of disjoint, mutually-exhaustive classes and return the class to which x_i belongs. We know that it is also referred to as generalization now. (2)Value Distortion: return $x_i + r$ instead of x_i. It is also referred to as randomization. (3)value dissociation: a value returned for a field of a record is a true value, but from the same field in some other record. This method is essentially the permutation and swapping approach. It is a global method and requires knowledge of values in other records.

6.2.3 Macrodata/Microdata Protection

Macrodata and microdata

Previously data is released in pre-aggregated tabular form through statistical database [11]. Such form of data are generally called *macrodata*, which represents estimated values of statistical characteristics concerning a given population. There are two main approaches for protecting statistical databases. The first approach restricts the statistical queries that can be made or the data that can be published. The second approach modifies the query result returned to users. The modification can be enforced directly on the stored data or at run time when computing the query results.

To accommodate the increasing demands for flexibility and availability of information from the users, *microdata* are to be released in many situations. Simple de-identification is not enough to protect privacy, as we have already seen in chapter 2. Quasi-identifiers linked with public data database can often lead to the disclosure of identities. Many approaches have been proposed to protect the privacy of microdata, which we will elaborate below. It shall be pointed out that our approach also falls in this category.

Microdata protection

The privacy vulnerability of the release of de-identified microdata was first discussed by Sweeney [1, 2]. It has been shown that, after linking a de-identified medical database with voter registration records, medical records of many individuals can be uniquely identified. Sweeney further proposed k-anonymity as a model for protecting privacy of microdata. Domain generalization and record suppression have been introduced as two techniques to achieve k-anonymity [5].

In [20], Samarati presented a framework for generalization and suppression based k-anonymity, where the concept of generalization hierarchies was formally proposed. Given a predefined domain hierarchy, the problem of k-anonymity is thus to find the minimal domain 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 domain generalizations. The concept of ℓ-diversity is introduced by Machanavajjhala et al. in [3] to prevent attackers with background knowledge. In [7], distribution of sensitive attributes is first considered. Based on this, a more robust privacy measure (which we refer 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], Xiao et al. proposed to let individuals specify privacy policies about their own attributes.

It has been shown that the problem of general k-anonymity with suppression and arbitrary domain generalizations (instead of pre-defined generalization hierarchies) is NP-complete [44, 45, 46]. Several approximation algorithms have been proposed [47, 44]. Several other works investigate the characteristics of k-anonymity. For example, Aggarwal discusses the curse of dimensionality related to k-anonymity [48]. In particular, he shows that it is not possible to create even a 2-anonymous table in high dimensional space without considerable information loss. In [10], Yao et al. show that, when several microdata tables are disclosed, even if each of them satisfies k-anonymity, by pooling them together, k-anonymity may be violated. They further design algorithms to detect such violations. Zhong et al. [39] devise a protocol for obtaining k-anonymous tables in distributed environments.

To improve the quality of the anonymized data, recently much work has been done to efficiently compute minimal and optimal generalizations [40, 6]. In [40], Bayardo and Agrawal presented a general model of the problem of finding optimal generalization and suppressions to achieve k-anonymity. The model can accommodate a variety of cost metrics. Pruning techniques have been proposed to reduce the search space of optimal generalization and optimization. In the Incognito approach of [6], generalization hierarchies are explored in a vertical way. It first computes the minimal solution to k-anonymity in the generalization hierarchy for each quasi-identifier. These solutions are then combined to form the candidate generalizations for the domain hierarchies of quasi-identifier pairs. This process continues until a set of minimal domain generalizations are obtained for the full domains of quasi-identifiers.

All the above works focus on introducing less imprecise information to microdata. But their impact on the accuracy of aggregate queries have not been discussed. Recently Xiao et al. [21] 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 [22].

Graph data protection

Since year 2007, there has been considerable interest in anonymizing data which can be represented as a graph, with motivation coming from wanting to publish social network data. Backstrom *et al.* [4] consider attacks on publishing such data with identifiers removed (the "fully censored" case). They study both active attacks, in which the attacker is allowed to insert a number of nodes and edges into the graph before it is published, and passive, in which the attacker learns all the edges incident on a set of linked nodes. In both cases, the authors show that with high probability, the known subgraph can be located in the overall graph, and hence information can be learnt about connections between nodes. However, as here, *nothing* is learnt about connections between nodes that are *not* incident on edges known to the attacker.

Hay *et al.*[26] analyze what privacy is present inherently within the structure of typical social networks, by measuring how many nodes have similar or identical neighborhoods (based, e.g. on degrees of nearby nodes). This is similar to the attack we studied in Section 4.5. They analyze what additional privacy is gained by deleting and then randomly inserting up to 10% of edge, but observe as we do that such large scale modification can significantly alter graph properties. Zhou and Pei [49] define privacy so that each node must have k others with the same (one-step) neighborhood characteristics, and measure the cost as the number of edges added, and number of node label generalizations. Korolova et al. [50] analyze attacks in a different model, where the attacker can only "buy" information about the neighborhood of certain nodes. Zheleva and Getoor [51] study the effectiveness of machine learning techniques to infer sensitive links which have been erased, given a graph in which non-sensitive links have been anonymized. They consider a collection of anonymizations based on grouping nodes: randomly deleting some non-sensitive edges; reporting only the number of edges between groups (similar to Section 4.3.3); and simply reporting whether or not two groups have any edges. They do not consider our approach of retaining the graph structure but hiding the mapping from entities to nodes. Our work differs from prior work essentially because we focus on a different region of the privacy-utility tradeoff: we consider settings where releasing the unlabelled graph is permitted, but lacks utility, whereas prior work does not allow such release.

Also relevant is work which considers relations with many sensitive attributes, since such data is often effectively represented in graph form. Nergiz *et al.*[52] mention the shortcomings of representing and anonymizing bitmap representations of relational data, which we argue is also insufficient for graph data in Section 4.3.2. Closest to our work in setting is recent work by Ghinita *et al.* [53] on anonymizing sparse high-dimensional data (since a bipartite graph can be seen as defining such a sparse relation). Their approach is to extend known permutation based methods [22, 21] to improve utility. In their data, sensitive attributes are rare, so they can ensure at most one sensitive attribute in each group of k individuals; in contrast, in our setting, every attribute (association) is sensitive and so we cannot apply their method. Moreover, [53] does not consider graph properties of the data, which is vital for us to preserve in our setting. Work on ℓ-diversity briefly considers the issue of multiple sensitive attributes, and concludes that much larger groups would be needed to guarantee privacy [3]. The crucial difference that allows our techniques to succeed is that although we have a large number of sensitive attributes (e.g. all papers) in graph data, the graph is sparse, so most of these attributes take a non-sensitive value.

6.3 Future work

The performance and utility of traditional database has been studied extensively in the literature. However, very limited research has been done concerning the performance of anonymized data. In the long run, I would like to explore research areas of combining performance, utility and privacy in the releasing of public database. In recent years, many emerging applications also pose new and challenging privacy requirement in their corresponding data releasing. Example databases include the social network data, data collected from sensor networks or RFID, etc. I also plan to extend my research to such applications which needs special privacy treatment.

Bibliography

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

[2] Latanya Sweeney. K-Anonymity, A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

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

[4] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore are thou R3579X? Anonymized social networks, hiddern patterns and structural steganography. In *International Conference on World Wide Web (WWW)*, 2007.

[5] Latanya Sweeney. Achieving K-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

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

[7] Ninghui Li and Tiancheng Li. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *International Conference on Data Engineering (ICDE)*, 2007.

[8] X. Xiao and Y. Tao. Personalized Privacy Preservation. In *ACM Conference on Management of Data (SIGMOD)*, Chicago, June 2006.

[9] R. Wong, J. Li, A. Fu, and K. Wang. (α, k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *ACM SIGKDD*, 2006.

[10] Chao Yao, Sean Wang, and Sushil Jajodia. Checking for k-Anonymity Violiation by Views. In *International Conference on Very Large Data Bases*, Trondheim, Norway, August 2005.

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

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

[13] Alexandre V. Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy Preserving Mining of Association Rules. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

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

[15] T. Dalenius and S.P. Reiss. Data-swapping: A Technique for Disclosure Control. *Journal of Statistical Planning and Inference*, 6, 1978.

[16] S.E. Fienberg and J. McIntyre. Data Swapping: Variations on a Theme by Dalenius and Reiss. In *Privacy in Statistical Databases*, Barcelona, Catalonia, June 2004.

[17] R.A.Jr. 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.

[18] "Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei". Minimality attack in privacy preserving data publishing. In *VLDB*, Vienna,Austria, 2007.

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

[20] Periangela Samarati. Protecting Respondent's Privacy in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

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

[22] Qing Zhang, Nick Koudas, Divesh Srivastava, and Ting Yu. Aggregate Query Answering on Anonymized Tables. In *International Conference on Data Engineering (ICDE)*, 2007.

[23] J. Bennett and S. Lanning. The Netflix prize. In *KDDCup Workshop*, 2007.

[24] A. Narayanan and V. Shmatikov. How to break anonymity of the Netflix prize dataset. Technical Report Technical Report arXiv:cs/0610105v1, arXiv, 2006.

[25] D. J. Martin, D. Kifer, A. Machanavajjhala, and J. Gehrke. Worse-case background knowledge for privacy-preserving data publishing. In *IEEE International Conference on Data Engineering*, 2007.

[26] M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. Technical report.

[27] M. R. Garey and D. S. Johnson. Computers and intractability, a guide to the theory of NP-completeness.

[28] Qing Zhang, Nick Koudas, Divesh Srivastava, and Ting Yu. Using Generalization and Permutation to Disguise Distributions. submitted for conference publication.

[29] Graham Cormode, Divesh Srivastava, Ting Yu, and Qing Zhang. Anonymizing Bipartite Graph Data using Safe Groupings. submitted for conference publication.

[30] W3C. Platform for privacy preferences (p3p) project. http://www.w3.org/P3P/.

[31] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Authorization Language (EPAL 1.1) Specification, 2003. http://www.zurich.ibm.com/security/enterprise-privacy/epal.

[32] A. C. Yao. How to generate and exchange secrets. In *27th IEEE Symposium on Foundations of Computer Science*, 1986.

[33] Irit Dinur and Kobbi Nissim. Revealing Information while Preserving Privacy. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2003.

[34] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2003.

[35] J. Gehrke. Models and Methods for Data Privacy. In *ACM SIGMOD Conference*, Baltimore, MD, June 2005.

[36] Alexandre Evfimievski. Randomization in privacy preserving data mining. *SIGKDD Explor. Newsl.*, 4(2):43–48, 2002.

[37] Murat Kantarcioglu and Chris Clifton. Privacy-preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. In *Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[38] Jaideep Vaidya and Chris Clifton. Privacy Preserving Association Rule Mining in Vertically Partitioned Data . In *Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2002.

[39] S. Zhong, Z. Yang, and R. N. Wright. Privacy-enhancing k-anonymization of customer data. In *ACM Conference on Principles of Database Systems(PODS)*, 2005.

[40] Roberto J. Bayardo Jr. and Rakesh Agrawal. Data Privacy through Optimal k-Anonymization. In *Proceedings of 21st IEEE International Conference on Data Engineering*, Tokyo, Japan, April 2005.

[41] Massimo Marchiori et al. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, April 2002.

[42] Marc Langheinrich. A P3P Preference Exchange Language 1.0 (APPEL1.0), April 2002.

[43] S. Micali O. Goldreich and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, 1987.

[44] Adam Meyerson and Ryan Williams. On the Complexity of optimal k-Anonymity. In *Proceedings of the 23rd ACM Symposium on the Principles of Database Systems*, 2004.

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

[46] K. LeFevre, D. DeWitt, and R. Ramakrishnan. Multidimensional k-Anonymity. Technical Report 1521, University of Winsconsin, 2005.

[47] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing Tables. In *International Conference on Database Theory*, January 2005.

[48] Charu Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *International Conference on Very Large Data Bases*, Trondheim, Norway, August 2005.

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

[50] A. Korolova, R. Motwani, S. Nabar, and Y. Xu. Link privacy in social networks. In *ICDE*, 2008.

[51] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In *International Workshop on Privacy, Security and Trust in KDD (PinKDD)*, 2007.

[52] M. E. Nergiz, C. Clifton, and A. E. Nergiz. Multirelational k-anonymity. In *IEEE International Conference on Data Engineering*, 2007.

[53] G. Ghinita, Y. Tao, and P. Kalnis. On the anonymization of sparse high-dimensional data. In *ICDE*, 2008.