Fundamenta Informaticae, Vol. 74, No.4, (2007), pp.595-612

IOS Press

Privacy Preserving Database Generation for Database Application Testing

Xintao Wu[*]
Department of Computer Science
University of North Carolina at Charlotte

Yongge Wang
Department of Software and Information System
University of North Carolina at Charlotte

Songtao Guo
Department of Software and Information System
University of North Carolina at Charlotte

Yuliang Zheng
Department of Software and Information System
University of North Carolina at Charlotte

Abstract. Testing of database applications is of great importance. Although various studies have been conducted to investigate testing techniques for database design, relatively few efforts have been made to explicitly address the testing of database applications which requires a large amount of representative data available. As testing over live production databases is often infeasible in many situations due to the high risks of disclosure of confidential information or incorrect updating of real data, in this paper we investigate the problem of generating synthetic databases based on a-priori knowledge about production databases. Our approach is to fit the general location model using various characteristics (e.g., constraints, statistics, rules) extracted from a production database and then generate synthetic data using model learned. The generated data is valid and similar to real data in terms of statistical distribution, hence it can be used for functional and performance testing. As characteristics extracted may contain information which may be used by attackers to derive some confidential information about individuals, we present our disclosure analysis method which applies cell suppression technique for identity disclosure and perturbation for value disclosure analysis.

Keywords: Data Generation, Disclosure Analysis, Statistical Database Modeling

Address for correspondence: Xintao Wu, Department of Computer Science, University of North Carolina at Charlotte, 9201 University City Blvd. Charlotte, NC 28223, USA. Email: xwu@uncc.edu

[*]This work was supported in part by U.S. National Science Foundation CCR-0310974. Part of this work was presented in ISMIS 2005 [24]. This extended version contains how to conduct identity disclosure and value disclosure in detail.

1. Introduction

Database application software testing is by far the most popular activity currently used by developers or vendors to ensure high software quality. A significant issue in database testing consists in the availability of representative data. Although application developers usually have some local development data for local functional testing, however, this small amount of data can not fulfill the requirements of some testing phases such as performance testing where a large amount of data is needed. On the other hand, testing over live production databases is often infeasible in many situations due to the high risks of disclosure of confidential information or incorrect updating of real data. Hence, synthetic data generation becomes an appealing alternative.

Recently, the authors in [23] have proposed a general framework for privacy preserving database application testing and investigated the method of generating synthetic database based on a-priori knowledge (e.g., constraints, statistics, rules, etc.) about the current production database. In testing the functions of database applications, the generated data need to satisfy all the constraints (e.g., no-Null, uniqueness, referential integrity constraints, domain constraints, and semantic constraints) and business rules underlying the live data. In testing the performance of database applications, it will be imperative that the data is resembling real data in terms of statistical distribution since the statistical nature of the data determines query performance. In addition to valid (in terms of constraints and rules) and resembling real data (in terms of statistical distribution), the data generated need to preserve privacy, i.e., the generated data should not disclose any confidential information that database owners would not want to reveal. Since the synthetic database is generated from a-priori knowledge about the live database, it is important to preclude confidential information from a-priori knowledge.

In this paper, we investigate how to use the *general location model* to model an existing production database and how to use the model learned to generate a synthetic database. We examine in detail how to extract statistics and rules to estimate parameters of the general location model and how to resolve the potential disclosure of confidential information in data generation using model learned. This issue is related to, but not identical to, the widely recognized problem of privacy preserving data mining (e.g., [3]). In the situation we present, our disclosure analysis is conducted at model level instead of tuple level.

The contribution of this work is twofold. First, the context in this paper is how to build the general location model from characteristics extracted from the production databases. As the model learned is the only means to generate data for release, all confidential information which attackers can derive is guaranteed to be contained in those parameters. Second, we examine how to resolve the potential disclosure (both identity disclosure and value disclosure) of confidential information entailed in the general location model. As the search space of parameters is much smaller than the space of perturbed data, this approach is more effective and efficient. Furthermore, we extend the concept of a uni-variate confidence interval to a multi-variate confidence region to measure privacy and confidentiality for multiple confidential attributes simultaneously.

The remainder of the paper is structured as follows. In section 2, we revisit the general location model and present some known results about density contour of multi-variate normal distribution from statistics, which will be used in value disclosure analysis. We describe our privacy preserving data generation system in section 3 and present how to fit the general location model in section 4. In section 5, we present in detail how to conduct identity disclosure and value disclosure in terms of the general location model. We review the related work in section 6. In section 7, we draw conclusions and describe directions for future work.

2. Preliminaries

2.1. The General Location Model Revisited

Let $\mathcal{A} = \{A_1, A_2, \cdots, A_q\}$ denote a set of categorical attributes and $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_p\}$ a set of numerical ones in a table with n entries. Suppose A_j takes possible domain values $1, 2, \cdots, d_j$, the categorical data $\mathcal{W}$ can be summarized by a contingency table with total number of cells equal to $D = \prod_{j=1}^{q} d_j$. let $\mathbf{y} = \{y_d : d = 1, 2, \cdots, D\}$ denote the number of entries in each cell. Clearly $\sum_{d=1}^{D} y_d = n$. The general location model [20] is defined in terms of the marginal distribution of $\mathcal{A}$ and the conditional distribution of $\mathcal{Z}$ given $\mathcal{A}$. The former is described by a multinomial distribution on the cell counts $\mathbf{y}$,

$$\mathbf{y} \mid \pi \sim M(n, \pi) = \frac{n!}{y_1! \cdots y_D!} {\pi_1}^{y_1} \cdots {\pi_D}^{y_D}$$

where $\pi = \{\pi_d : d = 1, 2, \cdots, D\}$ is an array of cell probabilities corresponding to $\mathbf{x}$. For each cell d, where $d = 1, 2, \cdots, D$, defined by the categorical attributes $\mathcal{A}$, the numerical attributes $\mathcal{Z}$ are then modeled as a conditionally multi-variate normal as:

$$f(\mathbf{z}|\mathbf{d}) = \frac{1}{(2\pi)^{p/2} \mid \Sigma_d \mid^{1/2}} e^{-1/2(\mathbf{z}-\mu_\mathbf{d})' {\Sigma_d}^{-1}(\mathbf{z}-\mu_\mathbf{d})}$$

where p-dimensional vector $\mu_\mathbf{d}$ represents the expected value of the random vector $\mathbf{z}$ for cell d, and the $p \times p$ matrix $\mathbf{\Sigma_d}$ is its variance-covariance matrix. The parameters of the generation location model can be written as $\theta = (\pi, \mu, \Sigma)$. The maximum likelihood estimates of θ is as follows:

$$\begin{aligned} \hat{\pi}_d &= \frac{x_d}{n} \\ \hat{\mu}_d^{'} &= x_d^{-1} \sum_{i \in B_d} z_i^{'} \\ \hat{\Sigma} &= x_d^{-1} \sum_{i \in B_d} (z_i - \hat{\mu}_d)(z_i - \hat{\mu}_d)^{'} \end{aligned} \tag{1}$$

where $B_d = \{i : \mu_i = E_d\}$ is the set of all tuples belonging to cell d.

An Illustrating Example Table 1 shows a data example of Mortgage dataset of customers with n tuples. Let us assume SSN and Name are confidential information and should be marked out. The remaining attributes are grouped into two parts: categorical attributes and numerical attributes. Each categorical attribute has its own domains, e.g., Gender with two domain values {Male, Female}. The categorical part can be summarized by a 4-dimensional contingency table with total number of cells equal to $D = \mid Zip \mid \times \mid Race \mid \times \mid Age \mid \times \mid Gender \mid$, where $||$ denote the size of domain values. We use x_d, $d = 1, 2, \cdots, D$ to denote the number of tuples in each cell. In general, tuples contained in one given table is a subset of the cartesian product of some domains. In other words, some cells may not contain any tuple due to either integrity constraints or the limited sample size. In this paper, we assume the domains for each categorical attribute are fixed while the volume of data is changing.

The general location model assumes the numerical attributes (i.e., Balance, Income, InterestPaid) of tuples in each cell follow a multivariate normal distribution. For example, the Balance, Income and InterestPaid of customers in the cell {28223, $Asian$, 20, $Male$} follow a 3-variate normal distribution

Table 1. An Example of Mortgage Dataset

			Categorical				Numerical		
	SSN	Name	Zip	Race	Age	Gender	Balance	Income	InterestPaid
1			28223	Asian	20	Male	10k	85k	2k
2			28223	Asian	30	Female	15k	70k	18k
3			28262	Black	20	Male	50k	120k	35k
.			.	.	.	.	.	.	.
n			28223	Black	25	Male	80k	110k	15k

with parameter μ_1, Σ_1, where μ_1 is a 3-vector means and Σ_1 is a 3×3 covariance matrix while those in the cell $\{28223,\ Asian,\ 20,\ Female\}$ may follow a 3-variate normal distribution with different means and covariance matrix.

Here we would emphasize that it is feasible to model various data using the general location model although a group of data may follow some other distributions (e.g., Zipf, Poison, Gamma etc.) in practice [20]. As we can see we define multi-variate normal distribution for data at the finest level and data at higher levels can be taken as a mixture of multi-variate normal distributions, hence we can theoretically use a mixture of multi-variate normal distributions to model any other distributions.

It is straightforward to see we can easily generate a dataset when the parameters of general location model are given. Generally, it involves two steps. First, we estimate the number of tuples in each cell d and generate x_d tuples. All x_d tuples from this cell have the same categorical attribute values inherited from the cell location of contingency table. Second, we estimate the mean and covariance matrix for those tuples in this cell and generate numerical attribute values based on the multi-variate normal model. In general, some columns' distribution may be functionally dependent on those of other columns, hence we would like to derive an approximate joint distribution of those independent columns which is used to generate synthetic data for performance testing.

2.2. Density Contour of Multi-variate Normal Distribution

In this section we present some known results about density contour of multi-variate normal distribution from statistics. These results will be used to conduct value disclosure of the general location model.

Result 1. (Constant probability density contour)
([13], page 134) Let $\mathcal{Z}$ be distributed as $N_p(\mu, \mathbf{\Sigma})$ with $\mid \mathbf{\Sigma} \mid > 0$. Then, the $N_p(\mu, \mathbf{\Sigma})$ distribution assigned probability $1 - \alpha$ to the solid ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)' \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \leq \chi_p^2(\alpha)\}$, where $\chi_p^2(\alpha)$ denotes the upper (100α-th) percentile of the χ_p^2 distribution with p degrees of freedom. The ellipsoid is centered at μ and have axes $\pm c\sqrt{\lambda_i}\mathbf{e}_i$, where $c^2 = \chi_p^2(\alpha)$ and $\mathbf{\Sigma}\mathbf{e}_i = \lambda_i \mathbf{e}_i$, $i = 1, \cdots, p$.

The multi-variate normal density is constant on surfaces where the squared distance $(\mathbf{z} - \mu)' \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu)$ is constant c^2. The chi-square distribution determines the variability of the sample variance. Probabilities are represented by volumes under the surface over regions defined by intervals of the z_i values. The axes of each ellipsoid of constant density are in the direction of the eigenvectors of $\mathbf{\Sigma}^{-1}$ and their lengths are proportional to the the square roots of the eigenvalues (λ_i) of $\mathbf{\Sigma}$.

Result 2 shows the general result concerning the projection of an ellipsoid onto a line in a p-dimensional space.

Result 2. (Projection of Ellipsoid)
([13], page 203) For a given vector $\ell \neq \mathbf{0}$, and $\mathbf{z}$ belonging to the ellipsoid $\{\mathbf{z} : \mathbf{z}'\mathbf{A}^{-1}\mathbf{z} \leq c^2\}$ determined by a positive definite $p \times p$ matrix $\mathbf{A}$, the projection (shadow) of $\{\mathbf{z}'\mathbf{A}^{-1}\mathbf{z} \leq c^2\}$ on ℓ is $c\frac{\sqrt{\ell'\mathbf{A}\ell}}{\ell'\ell}\ell$ which extends from $\mathbf{0}$ along ℓ with length $c\sqrt{\frac{\ell'\mathbf{A}\ell}{\ell'\ell}}$. When ℓ is a unit vector, the shadow extends $c\sqrt{\ell'\mathbf{A}\ell}$ units, so $\mid \mathbf{z}'\ell \mid \leq c\sqrt{\ell'\mathbf{A}\ell}$.

3. System Overview

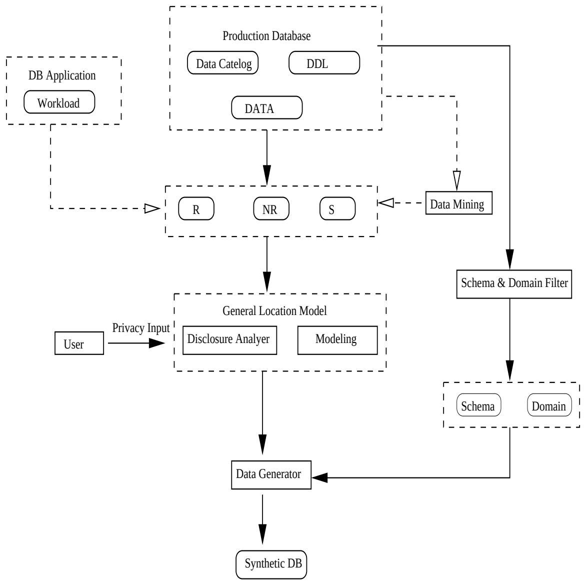

Figure 1. Architecture of data generation system

Figure 1 shows the architecture of synthetic database generation system. The system is intended to generate a synthetic database using various characteristics extracted from the production database for testing database applications, instead of generating benchmark for use in the performance evaluation of DBMS. We assume databases are based on the relational model in our paper. A relational database is a set of relation schemas together with a set of integrity constraints which restrict the possible values of the database states.

We would like automate the data generation process as much as possible. We use the same approach as in [4] to extract various constraint information from schemes which are defined by Data Definition Language (DDL). It is desirable that the generated data in synthetic databases also satisfy the constraints.

A major advantage is that our system can extract more complex characteristics (e.g., statistics and rules) in addition to constraints from data catalog and data when the underlying production database is available. As we discussed in introduction, even if one synthetic database satisfies all constraints, it does not mean it can fulfill users' testing requirement as it may have different data distribution than the production database. Hence our approach extracts characteristics (e.g., statistics, rules) from the production database, fits the general location model using characteristics extracted, and generates a synthetic database using model learned. As shown in Figure 1, the characteristics of production databases can be extracted from three parts: DDL, Data Dictionary, and Data. In order to ensure that the data is close looking or statistically similar to real data, or at least from the point of view of application testing, we need to have the statistical descriptions, $\mathcal{S}$, and non-deterministic rules, $\mathcal{NR}$, of real data in production databases. These two sets describe the statistical distributions or patterns of underlying data and may affect the size of relations derived as a result of the evaluations of queries the application will need to execute. Our intuition is that, for database applications, if two databases are approximately the same from a statistical viewpoint, then the performance of the application on the two databases should also be approximately the same [1]. Furthermore, our system includes one disclosure analysis component which helps users remove those characteristics which may be used by attackers to derive confidential information in the production database. As all information used to generate synthetic data in our system are contained in characteristics extracted, our disclosure analysis component can analyze and preclude the potential disclosure of confidential information at the characteristics (i.e., statistics and rules) level instead of data level.

4. Database Modeling via the General Location Model

4.1. Extracting characteristics from data dictionary

Data dictionary consists of read-only base tables that store information about the database. When users execute an SQL command (e.g., CREATE TABLE, CREATE INDEX, or CREATE SEQUENCE) to create an object, all of the information about column names, column size, default values, constraints, index names, sequence starting values, and other information are stored in the form of metadata to the data dictionary. Most commercial DBMSs also collect statistical information regarding the distribution of values in a column to be created. This statistical information can be used by the query processor to determine the optimal strategy for evaluating a query. As the data in a column changes, index and column statistics can become out-of-dated and cause the query optimizer to make less-than-optimal decisions on how to process a query. SQL server automatically updates this statistical information periodically as the data in the tables changes. In our system, we have one component which simply accesses tables in data dictionary and fetch characteristics related.

4.2. Extracting characteristics from data

The statistics information about columns extracted directly from data dictionary is usually with high granularity which may be insufficient to derive relatively accurate model. In practice it is usually true that users can collect more statistics at low granularity from original data or a sample of real data themselves.

[1]Here we assume that file organizations, sorted fields, and index structures of the production database x are not private information and the synthetic data generator use these information to build the synthetic database in the same way that production database has been built.

```
SELECT     Zip, Race, Age, Gender, COUNT(*),
           AVG(Balance), AVG(Income), AVG(InterestPaid),
           VAR_POP(Balance), VAR_POP(Income), VAR_POP(InterestPaid),
           COVAR_POP(Balance,Income), COVAR_POP(Balance,InterestPaid),
           COVAR_POP(Income,InterestPaid)
FROM       Mortgage
GROUP BY   Zip, Race, Age, Gender
HAVING     COUNT(*) > 5
```

Figure 2. Example of extracting statistics using SQL

```
Q1: SELECT AVG(Income), AVG(InterestPaid) FROM Mortgage WHERE Zip = z AND Race = r
Q2: SELECT AVG(Income) FROM Mortgage WHERE Age = a AND Zip = z
```

Figure 3. Workload example of Mortgage database

Figure 2 presents one SQL command to extract statistics at the finest granularity level. It returns all information needed to derive parameters of the general location model. For example, the value from aggregate function COUNT(*) is the estimate, x_d, of the number of tuples in each cell while the values from aggregate functions (i.e., AVG, VAR_POP, COVAR_POP) are the estimates of mean vectors and covariance matrices respectively of the multi-variate normal distribution of each cell. It is worth pointing out that all aggregate functions can be used with GROUP BY clause with CUBE or ROLLUP option if we want to extract statistics at all possible granularity.

In practice, it may be infeasible to extract statistics at the finest level because statistics at the finest level may contain too much information and may be exploited by attackers to derive some confidential information about production databases. As our goal is to generate synthetic data for database application testing, we may extract statistics which are only related to queries in workload of database software. For the workload which contains two queries shown in Figure 3, it is clear that the distribution of underlying

```
Statistics 1: SELECT     Zip, Race, COUNT(*), AVG(Balance), AVG(InterestPaid)
              FROM       Mortgage
              GROUP BY   Zip, Race
              HAVING     COUNT(*) > 5

Statistics 2: SELECT     Zip, Age, COUNT(*), AVG(Income)
              FROM       Mortgage
              GROUP BY   Zip, Age
              HAVING     COUNT(*) > 5
```

Figure 4. SQLs of extracting statistics for workload

Rule 1: IF Zip = 28223, Race = Asian, and Age in (25, 40) THEN Balance is in (20k,30k)
with support s = 900 and confidence c= 90 %.

Rule 2: IF Zip = 28262 THEN Race = White with support s = 5000 and confidence c = 80 %

Figure 5. Example of non-deterministic rules for Mortgage dataset

data at some high level (instead of at the finest level) is sufficient to capture the relation with the execution time of queries in workload. For example, the approximate distribution on $Zip, Race$ would satisfy the performance requirements of Q1 in workload. In this case, we may only extract statistics necessary for query performance of queries. Figure 4 presents two SQL commands to extract statistics for two queries shown in Figure 3.

4.3. Extracting characteristics from rule sets

To derive the deterministic rule set $\mathcal{R}$, we take advantage of the database schema, which describes the domains, the relations, and the constraints the database designer has explicitly specified. Some information (function dependencies, correlations, hierarchies etc.) can be derived from database integrity constraints such as foreign keys, check conditions, assertions, and triggers. Furthermore, users may apply some data mining tools to extract non-deterministic rules $\mathcal{NR}$ from the production database. The non-deterministic rule set $\mathcal{NR}$ helps describe the statistical distributions or patterns of underlying data and may affect the size of relations derived as a result of the evaluations of queries the application will need to execute. Formally, each rule in $\mathcal{R}$ and $\mathcal{NR}$ can be represented as a declarative rule and is generally of the form:

IF <*premise*> THEN <*conclusion*> [with support s and confidence c].

The rules may include exact, strong, and probabilistic rules based on the support and confidence. We note here that complex predicates and external function references may be contained in both the condition and action parts of the rule. Anyone with subject matter expertise will be able to understand the business logic of the data and can develop the appropriate conditions and actions, which will then form the rule set.

Figure 5 shows two examples of non-deterministic rules for the Mortgage database. We can interpret Rule 1 as there are 1000 customers with Zip = 28223, Race = Asian, and Age in (25, 40) and 90 % of them with Balance in the range of (20k, 30k). It is straightforward to see that these rules can be mapped to statistics of the general location model at some granularity. For example, the number of data entries in cell (28223, Asian, 25-40, All) is 1000 and we can derive average balance of data entries in this cell from the clause Balance in (20k,30k) with confidence c= 90 %.

4.4. Fitting model using characteristics

It is easy to see all characteristics (i.e., $\mathcal{S}$, $\mathcal{R}$, $\mathcal{NR}$) extracted from production database can be mapped to constraints of parameters of the general location model at the finest level. Given those constraints, we can apply linear programming techniques to derive parameters of the general location model at the finest level. However, it is infeasible to apply linear programming techniques directly in practice due to high complexity (the number of variables is linear of the number of cells D while the number of constraints is large). As we know, the problem of estimating the cell entries at the finest granularity subject to some linear constraints is known to be NP-hard. In our system, we combine some heuristics to derive parameters from high level constraints. For example, from Rule 1, we can initialize the number of tuples in those 30 cells (28223, Asian, Age, Gender) where Age is in (25,40) and Gender in $\{$Male, Female$\}$ as 30 ($\frac{900}{15\times2}$) when we assume the tuples are uniformly distributed among Age and Gender in cell (28223, Asian, Age, Gender).

5. Disclosure Analysis

Disclosures which can occur as a result of inferences by attackers include two classes: identity disclosure and value disclosure. Identity disclosure relates to the disclosure of the identity of an individual in the database while value disclosure relates to the disclosure of the value of a certain confidential attribute of that individual.

Given characteristics $\mathcal{DB} = \{\mathcal{S} \cup \mathcal{R} \cup \mathcal{NR}\}$ and a set of confidential information $\bar{\mathcal{DB}} = \{\bar{S} \cup \bar{R} \cup \bar{NR}\}$, our initial problem is to find a $\hat{\mathcal{DB}}$ such that 1) $\hat{\mathcal{DB}} \subseteq \mathcal{DB}$ and 2) no confidential information in $\bar{\mathcal{DB}}$ can be entailed from $\hat{\mathcal{DB}}$ within a given bound. We can see this initial problem is very complex as rules and statistics have different formats. In our system, we use the general location model, which is built from rules and statistics, to generate the final data. So all the information are contained in the parameters of the general location model, i.e., $\theta = (\pi, \mu, \Sigma)$.

From section 2.1, we know 1) π is only used for multinomial distribution and can be estimated using $\hat{\pi}_d = \frac{x_d}{n}$, where x_d is the number of entries in cell d; and 2) μ, Σ is only used for multi-variate normal distribution of numerical attributes for those entries in a given cell d. As π_d is only related to categorical attributes and μ, Σ are only related to numerical attributes, we can analyze them separately. In the remainder of this subsection, we discuss in detail how to check whether π_d incurs identity disclosure and whether μ, Σ incurs value disclosure.

5.1. Identity Disclosure

During the data generation process, data have been de-identified by suppressing SSN and Name so not to disclose the identities of the individuals to whom the data refer. However, values of other released attributes, such as Zip, Race, and Age can also appear in some external table (e.g., voter list) jointly with the individuals' identities, and can therefore allow them to be tracked. Several microdata disclosure protection techniques have been developed in the context of statistical databases [1]. Recently, Samarati [19] introduced the definition of quasi-identifiers as attributes that can be exploited for linking and of k-anonymity as characterizing the degree of data protection with respect to inference by linking.

In order to preserve the confidentiality of individuals, we typically will not release any rule which involves few records. However, attackers may still be able to derive or estimate the values of some confidential cells by analyzing some cells from the released characteristics. In our scenario, confidential information may even exist at aggregate levels. For example, in the table which records the number of patients visiting physicians to receive treatments, the information on Patient-Doctor and Doctor-Treatment are not sensitive and are publicly accessible. However, the Patient-Treatment information is sensitive, and so, confidential. This problem is referred as determining upper and lower bounds on the cells of the cross-classification given a set of margins [7, 6]. Upper and lower bounds induced by some fixed set of marginal on the cell entries of a contingency table are of great importance in measuring the disclosure risk associated with the release of these marginal totals. If the induced upper and lower bounds are too tight or too close to the actual sensitive value in a confidential cell entry, the information associated with that cell may be disclosed.

In our system, from characteristics $\mathcal{DB} = \{\mathcal{S} \cup \mathcal{R} \cup \mathcal{NR}\}$, we extract a set of cells, $\mathcal{C}^0$, and the number of entries in each cell $c \in \mathcal{C}^0$. From a list of private rules and statistics, $\bar{\mathcal{DB}} = \{\bar{S} \cup \bar{R} \cup \bar{NR}\}$, we similarly extract a list of confidential cells, $\mathcal{C}^1$. For each confidential cell $c \in \mathcal{C}^1$, a confidential range $[x_c^l, x_c^u]$ which contains the true value of the number of entries, x_c, is derived. $[x_c^l, x_c^u]$ here denotes the confidential range which database owner does not want attackers to predict. It is clear that predicting confidential value within a smaller confidential range constitutes compromise. Now our identity disclosure problem is to find a set of cells, $\mathcal{C}^2$, which can be released for data generation, such that 1) $\mathcal{C}^2 \subseteq \mathcal{C}^1$ and 2) no confidential information x_c ($c \in \mathcal{C}^1$) can be predicted in range $[x_c^l, x_c^u]$ from the information contained in $\mathcal{C}^2$. As this problem is NP-hard, in our system we apply similar heuristics as presented in [9] to remove confidential information contained in $\mathcal{C}^1$ one by one. Basically, it identifies those cells contained in $\mathcal{C}^0$ which need to be suppressed in order to hide the specific confidential information in $\mathcal{C}^1$. We present the details in Appendix A.

5.2. Value Disclosure

From Result 1, we know the ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \le \chi_p^2(\alpha)\}$, which is yielded by the paths of $\mathbf{z}$ values, contains a fixed percentage, $(1 - \alpha)100\%$ of customers. Although snoopers may use various techniques to estimate and predict the confidential values of individual customers, however, all confidential information which snoopers can learn is the bound of ellipsoid in our scenario.

In [3], privacy is measured in terms of confidence intervals for each single numerical attribute. Given confidence $c\%$, for each randomized value z, an interval $[z - w_1, z + w_2]$ is defined such that for all nonrandomized values x,

$$P[z - w_1 \le x \le z + w_2 \mid z = x + y, y \sim F_y] \ge c\%$$

The shortest width $w = w_1 + w_2$ for a confidence interval is used as the amount of privacy at $c\%$ confidence level. In the p-dimensional space, an $c\%$ confidence region will be an ellipsoidal region given by its probability density contour. This region consists of values of $\mathbf{x}$ (i.e., a vector over all numerical attributes) that may be accepted at the $1 - c\%$ level of significance.

Assume $\mathcal{E}$ is the ellipsoid from the original data $\mathbf{z}$ at one given confidence level $1 - \alpha$ and $\hat{\mathcal{E}}$ is the ellipsoid from the modified distribution (or generated data). Equation 2 defines the measure of disclosure of $\mathbf{z}$ when $\hat{\mathbf{z}}$ is given.

$$D(\mathbf{z}, \hat{\mathbf{z}}) = \frac{\mid vol(\mathcal{E} \cap \hat{\mathcal{E}}) \mid}{\mid vol(\mathcal{E} \cup \hat{\mathcal{E}}) \mid} \tag{2}$$

Here compromise is said to occur when $D(\mathbf{z}, \hat{\mathbf{z}})$ is greater than τ, specified by the database owner. The greater the $D(\mathbf{z}, \hat{\mathbf{z}})$, the closer the estimates are to the true distribution, and the higher the chance of disclosure. In other words, if the ellipsoid learned by snoopers is close enough to that specified by database owners, we say partial disclosure occurs. To compute the volume of density contour, we have the following results as shown in Proposition 5.1.

Proposition 5.1. (Volume of density contour)
The volume of an ellipsoid $\{\mathbf{z} : (\mathbf{z}-\mu)^{'}\mathbf{\Sigma}^{-1}(\mathbf{z}-\mu) \leq \chi_p^2(\alpha)\}$ is given by $vol(\mathcal{E}) = \eta(\sqrt{\chi_p^2})^p \mid \Sigma^{1/2} \mid$ or $vol(\mathcal{E}) = \eta(\sqrt{\chi_p^2})^p \prod_{i=1}^p \sqrt{\lambda_i}$, where η is the volume of the unit ball in $\mathbf{R}^p$, and λ_i is the i-th eigenvalue of matrix $\mathbf{\Sigma}$.

Proof. From [11], we know the volume of an ellipsoid $\{\mathbf{z} : (\mathbf{z}-\mu)^{'}\mathbf{A}^{-1}(\mathbf{z}-\mu) \leq 1\}$ determined by one positive definite $p \times p$ matrix $\mathbf{A}$ is given by $vol(\mathcal{E}) = \eta \mid \mathbf{A^{1/2}} \mid$, where η is the volume of the unit ball in $\mathbf{R}^p$. We replace $\mathbf{A}$ with $\frac{\mathbf{\Sigma}^{-1}}{\chi_p^2(\alpha)}$, then we get $vol(\mathcal{E}) = \eta(\sqrt{\chi_p^2})^p \mid \Sigma^{1/2} \mid$.

From spectral decomposition $\mathbf{\Sigma} = \sum_{i=1}^p \lambda_i \mathbf{e_i}\mathbf{e_i}^{'} = \mathbf{P\Lambda P}^{'}$, we get $\mid \mathbf{\Sigma} \mid = \mid \mathbf{P} \mid\mid \mathbf{\Lambda} \mid\mid \mathbf{P}^{'} \mid = \mid \mathbf{P} \mid\mid \mathbf{P}^{'} \mid\mid \mathbf{\Lambda} \mid = \mid \mathbf{PP}^{'} \mid\mid \mathbf{\Lambda} \mid$. As $\mathbf{PP'} = \mathbf{I}$, then we have $\mid \mathbf{\Sigma} \mid = \mid \mathbf{\Lambda} \mid$. Due to $\mid \mathbf{\Sigma}^{1/2} \mid = \mid \mathbf{\Lambda} \mid^{1/2} = \prod_{i=1}^p \sqrt{\lambda_i}$, hence we have $vol(\mathcal{E}) = \eta(\sqrt{\chi_p^2})^p \prod_{i=1}^p \sqrt{\lambda_i}$.

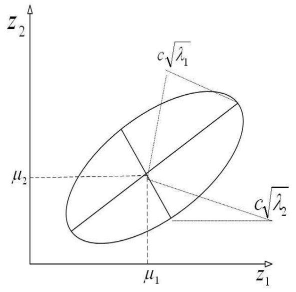

Figure 6. A constant density contour for a bi-variate normal distribution

Figure 6 shows one constant density contour containing 95% of the probability under the ellipse surface for one bi-variate $\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$, which follows a bi-variate normal distribution $N(\mu, \Sigma)$ with $\mu = \begin{pmatrix} \mu_\mathbf{1} \\ \mu_\mathbf{2} \end{pmatrix}$ and $\mathbf{\Sigma} = \begin{pmatrix} \sigma_\mathbf{11} & \sigma_\mathbf{12} \\ \sigma_\mathbf{21} & \sigma_\mathbf{22} \end{pmatrix}$. $\lambda = \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$ is the eigenvalues of covariance matrix $\mathbf{\Sigma}$ and two axes have length of $c\sqrt{\lambda_1}$ and $c\sqrt{\lambda_2}$ respectively, here $c = 2.45$ as $\sqrt{\chi_2^2(0.05)} = \sqrt{5.99} = 2.45$. We can see the major axis of ellipse is associated with the largest eigenvalue (λ_1). The size of this ellipse is given by $5.99\sqrt{\lambda_1 \times \lambda_2}$, as $\chi_2^2(0.05) = 5.99$.

However, to evaluate the measure of disclosure, $D(\mathbf{z} \mid \hat{\mathbf{z}})$, as shown in Equation 2, we need to compute the volume of the intersection (or union) of two ellipsoids. This problem is shown as NP-hard and some approximation techniques were surveyed in [12]. One heuristic we apply here is to use a hyper-rectangle (e.g., Bonferroni's rectangle or Roy's rectangle [13]) to approximate the ellipsoid. As we know, computing the intersection (or union) of two hyper-rectangle in high dimensional space is straightforward.

Although we can easily derive the ellipsoid $\mathcal{E}$ (or $\hat{\mathcal{E}}$) from the original (modified) data or distribution, it may be hard for database owners to specify privacy requirements using the ellipsoid.

$$d(z, \hat{z}) = \frac{[z^l, z^u] \cap [\hat{z}^l, \hat{z}^u]}{[z^l, z^u] \cup [\hat{z}^l, \hat{z}^u]} \tag{3}$$

Equation 3 defines the measure of disclosure for one confidential attribute. Here the confidence interval $[z^l, z^u]$ is specified by the database owner for each confidential attribute z. In this case, we can conduct disclosure analysis by comparing the best confidence interval, $[\hat{z}^l, \hat{z}^u]$, derived by snoopers with the confidence interval, $[z^l, z^u]$, specified by the database owner. To compute the projection of one ellipsoid on each axis, we have the following results as shown in Proposition 5.2.

Proposition 5.2. (Simultaneous Confidence Intervals)
Let $\mathbf{Z}$ be distributed as $N_p(\mu, \mathbf{\Sigma})$ with $\mid \mathbf{\Sigma} \mid > 0$. The projection of this ellipsoid $\{\mathbf{z} : (\mathbf{z} - \mu)^{'} \mathbf{\Sigma}^{-1} (\mathbf{z} - \mu) \leq \chi_p^2(\alpha)\}$ on axis $\mathbf{z_i} = (0, \cdots, 1, \cdots, 0)^{'}$ (only the i-th element is 1, all other elements are 0) has bound:

$$[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$$

Proof. From Result 2, we know the projection of an ellipsoid $\{\mathbf{z} : \mathbf{z}^{'} \mathbf{A}^{-\mathbf{1}} \mathbf{z} \leq c^2\}$ on a given unit vector ℓ has length $len = c\sqrt{\ell^{'} \mathbf{A} \ell}$. We replace $\mathbf{A}$ with

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & . & . & . & \sigma_{1p} \\ \sigma_{12} & \sigma_{22} & . & . & . & \sigma_{2p} \\ . & . & . & . & . & . \\ \sigma_{i1} & \sigma_{i2} & & \sigma_{ii} & & \sigma_{ip} \\ . & . & & & . & . \\ \sigma_{p1} & \sigma_{p2} & . & . & . & \sigma_{pp} \end{pmatrix}$$

, replace ℓ as $\mathbf{z_i} = (0, \cdots, 1, \cdots, 0)^{'}$, and replace c as $\sqrt{\chi_p^2(\alpha)}$, then we get the length of projection as $len = \sqrt{\chi_p^2(\alpha)\sigma_{ii}}$. Considering the center of this ellipsoid, we have the bound as $[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$.

It is easy to see from Proposition 5.2 that the confidence interval for each attribute (by projecting on each axis) is only dependent on μ_i, σ_{ii} while it is independent with covariance values σ_{ij}, where $i \neq j$.

To check whether a given distribution of $\mathbf{z}$ may incur value disclosure for one attribute z, we can simply compare the disclosure measure $d(z, \hat{z})$ with τ, specified by the database owner. If disclosure occurs, we need to modify parameters $\mu, \mathbf{\Sigma}$. As we know from Proposition 5.2, the mean vector μ determines the center of ellipsoid or the center of projection interval while the covariance matrix $\mathbf{\Sigma}$ determines the size of ellipsoid or the length of projection interval. As the change of μ will significantly affect the data distribution (it will affect the accuracy of analysis or mining subsequently), in the remainder of this paper we focus only on how to change variance matrix $\mathbf{\Sigma}$ to satisfy user's security requirement. From the bound $[\mu_i - \sqrt{\chi_p^2(\alpha)\sigma_{ii}}, \quad \mu_i + \sqrt{\chi_p^2(\alpha)\sigma_{ii}}]$, we can easily derive σ_{ii} to satisfy privacy requirements on the confidential attribute z.

Please note that both the ellipsoid $\mathcal{E}$ and the confidence interval $[z^l, z^u]$ from discussions above are specified for a group of customers which are modeled by one multi-variate normal distribution with the same parameters. Hence both $\mathcal{E}$ and $[z^l, z^u]$ are privacy specifications at the aggregate level. In practice, each customer j may specify his own privacy interval $[z^l_{(j)}, z^u_{(j)}]$ which contains his confidential value $z_{(j)}$. In this scenario, the database owner is required to prevent snoopers to derive or estimate the confidential value falling into its privacy interval. We use $[\hat{z}_l, \hat{z}_u]$ to denote the numerical attribute z's confidence interval learned by snoopers through projecting the ellipsoid on its axis. If the derived confidence interval $[\hat{z}_l, \hat{z}_u]$ by snoopers is close to the customer j's privacy interval $[z^l_{(j)}, z^u_{(j)}]$, we say *individual value disclosure* occurs for customer j. Currently we are working on how to evaluate this individual value disclosure in database modeling.

6. Related Work

Testing of database applications is of great importance since undetected faults in these applications may result in incorrect modification or accidental removal of crucial data. Although various studies have been conducted to investigate testing techniques for database design, relatively few efforts have been made to explicitly address the testing of database applications. The problem of database application testing can be categorized into three parts: database generation, input test cases preparation and test outcomes verification. In this paper, we focus on database generation.

There have been some prior investigations into data generation. For example, Transaction Processing Performance Council has released a dozen of TPC Benchmarks and many researchers have evaluated those Benchmarks (e.g., [10, 14, 16]). There are also some other data generation tools (e.g., [17, 15]) available. However, both TPC Benchmarks and other data generation tools are built for assessing the performance of database management systems, rather than for testing complex real world database applications. They lack the required flexibility to produce more realistic data needed for application testing, i.e., the generated data also need to satisfy all the constraints and business rules underlying the live data.

To generate realistic data for database applications, the authors in [4, 5] investigate how to populate the database with meaningful data that satisfy database constraints. They present a tool which inputs a database schema definition, along with some additional information from the user, and outputs a valid database state. The tool can handle not-NULL, uniqueness, referential integrity constraints, and some domain constraints and semantic constraints. Most constraints are included in data schemas which are expressed by SQL data definition language (DDL). The tool parses the schema definition for the database underlying the application to be tested using PostgreSQL (http://www.postgresql.org), then collects relevant information about tables, attributes, and constraints from the parse tree. The generation technique was motivated by the category-partition testing technique.

The inherent challenge of generating data for database applications is the tradeoff between similarity and privacy preservation. If the data is too synthetic (e.g., completely uniform distributions), it runs the risk of being rejected for not capturing he interesting patterns of a real data set. Conversely, if it employs data from the real world directly, it risks the violation of privacy issues. In terms of performance testing, using a large amount of *resembling* data is necessary to guarantee its satisfied performance when software is deployed. The generated data need to resemble real data in terms of statistical distribution in order to fulfill requirements of applications testing. The authors in [21] points out the importance of providing meaningful, representative data with realistic skew, sparsity and data distributions for benchmarking database system performance. Zheng et al. in [25] show that artificial data sets have very different characteristics from the real-world data sets and hence there is a great need to use real-world data sets as benchmarks for association rule mining. The authors in [22] empirically evaluate the effect of data distribution on workload performance using TPC-C and TPC-H benchmarks and show the importance of generating statistically similar synthetic data for performance testing.

As many databases maintain data on sensitive or confidential information such as income and assets for real customers, it is imperative to guarantee the data generated can not disclose any private or confidential information. The field of statistical databases [1, 8] has developed various methods to prevent the disclosure of confidential individual data while satisfying requests for aggregate information. The proposed techniques can be broadly classified into query restriction and data perturbation [3]. As our aim here it to release data for application testing, query restriction techniques are not feasible in our scenario.

There are various approaches to assessing risk of identity disclosure and most of them relate to the inadvertent release of small counts in the full k-way table [7, 8]. We should point out that the privacy consideration in the current literature for statistical database is not enough for many general environments which contain both categorical and numerical attributes. In most statistical database literatures, the privacy concerned is about the re-identification of some specific entries in the database.

The objective of randomized based privacy-preserving data mining [2, 3, 18] is to prevent the disclosure of confidential individual values while preserving general patterns and rules. The idea of these randomization based approaches is that the distorted data, together with the distribution of the random data used to distort the data, can be used to generate an approximation to the original data values while the distorted data does not reveal private information, and thus is *safe* to use for mining. One major challenge for current approaches is how to evaluate privacy breaches effectively as the perturbed data space which is used for disclosure analysis is almost infinite.

7. Conclusion and Future Work

In this paper we investigated how to generate synthetic databases using the general location model which is built using various characteristics extracted from production databases. We also investigated how to conduct disclosure analysis on the general location model which is used to generate synthetic data. Our synthetic database generated has similar distributions or patterns as the production database while preserving privacy, hence it can be used for database application testing. There are some aspects of this work that merit further research. Among them, we are trying to figure out how to better screen out confidential information from released characteristics, especially when linear combinations exist among numerical attributes. We will also conduct a complete study on how different data distributions affect workload performance using various datasets. Another area for future work is centered on refining the architecture of the data generator itself. This could include changes to allow further use of real world data sources (e.g., historical data) for increased realism and more rapid adjustment to emerging data trends or perturbation.

References

[1] Adam, N. R., Wortman, J. C.: Security-control methods for statistical databases, *ACM Computing Surveys*, **21**(4), Dec 1989, 515–556.

[2] Agrawal, D., Agrawal, C.: On the design and quantification of privacy preserving data mining algorithms, *Proceedings of the 20th Symposium on Principles of Database Systems*, 2001.

[3] Agrawal, R., Srikant, R.: Privacy-preserving data mining, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, May 2000.

[4] chays, D., Dan, S., Frankl, P., Vokolos, F., Weyuker, E.: A framework for testing database applications, *Proceedings of the ISSTA*, Portland, Oregon, 2000.

[5] Chays, D., Deng, Y., Frankl, P., Dan, S., Vokolos, F., Weyuker, E.: AGENDA: a test generator for relational database applications, Technical report, Polytechnic University, 2002.

[6] Dobra, A., Fienberg, S. E.: Bounds for cell entries in contingency tables given marginal totals and decomposable graphs, *PNAS*, **97**(22), 2000, 11885–11892.

[7] Dobra, A., Fienberg, S. E.: Bounds for cell entries in contingency tables induced by fixed marginal totals with applications to disclosure limitation, *Statistical Journal of the United Nations ECE*, **18**, 2001, 363–371.

[8] Domingo-Ferrer, J.: Current directions in statistical data protection, *Proceedings of the Statistical Data Protection*, 1998.

[9] Fagan, J.: Cell suppression problem formulations- exact solution and heuristics, August 2001.

[10] Gray, J., Sundaresan, P., Englert, S., Baclawaski, K., Weinberger, P. J.: Quickly generating billion-record synthetic databases, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 1994.

[11] Grotschel, M., Lovasz, L., Schrijver, A.: *Geometric algorithms and combinatorial optimization*, Springer, New York, 1988.

[12] Henrion, D., Tarbouriech, S., Arzelier, D.: LMI approximations for the radius of the intersection of ellipsoids: a survey, *Journal of Optimization Theory and Applications*, **108**(1), 2001.

[13] Johnson, R., Wichern, D.: *Applied Multivariate Statistical Analysis*, Prentice Hall, 1998.

[14] Leutenegger, S., Dias, D.: A modeling study of the TPC-C Benchmark, *Proceedings of the ACM SIGMOD Conference on Management of Data*, Washington, D.C., May 1993.

[15] Niagara: http://www.cs.wisc.edu/niagara/datagendownload.html.

[16] Poess, M., Stephens, J.: Generating thousand benchmark queries in seconds, *Proceedings of the 30th VLDB Conference*, 2004.

[17] Quest: http://www.quest.com/datafactory.

[18] Rizvi, S., Haritsa, J.: Privacy preserving association rule mining, *Proceedings of the 28th International Conference on Very Large Data Bases*, August 2002.

[19] Samarati, P.: Protecting respondents' identities in microdata release, *IEEE Transaction on Knowledge and Data Engineering*, **13**(6), 2001, 1010–1027.

[20] Schafer, J.: *Analysis of Incomplete Multivariate Data*, Chapman Hall, 1997.

[21] Stephens, J., Poess, M.: MUDD: A multi-dimensional data generator, *Proceedings of the 4th International Workshop on Software and Performance*, 2004.

[22] Wu, X., Sanghvi, C., Wang, Y., Zheng, Y.: Privacy aware data generation for testing database applications, *Proc. of the 9th International Database Engineering and Application Symposium*, July 2005.

[23] Wu, X., Wang, Y., Zheng, Y.: Privacy preserving database application testing, *Proceedings of the ACM Workshop on Privacy in Electronic Society*, 2003.

[24] Wu, X., Wang, Y., Zheng, Y.: Statistical database modeling for privacy preserving database generation, *Proc. of the 15th International Symposium on Methodologies for Intelligent Systems*, May 2005.

[25] Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rule algorithms, *Proceedings of the 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, August 2001.

A. Cell Suppression

Several cell suppression problem formulations have been given in [9]. It is not directly applicable to our questions since some values in the original contingency table are unknown in our case. In the following, we modify these formulations so that they could be used in our case.

Let $a = \{a_i\}_{i=1}^{n}$ be a contingency table. Some of these values a_i may be undefined. If a_i is undefined, then we write $a_i = *$. Let H be an m by n matrix describing the linear constraints on the feasible contingency table. That is, a feasible contingency table y should satisfy

$$\begin{aligned} Hy &= 0 \\ y &\geq 0. \end{aligned}$$

Let $P = \{i_1, \ldots, i_p\} \subset \{1, \ldots, n\}$ be the original specified suppressions. For each $i_k \in P$, there is a lower bound requirement l_k and an upper bound requirement u_k. The interpretation of these bounds is as follows. For each $i_k \in P$, the dataset owner does not want the public to infer from the published data that the value of c_{i_k} lies in $(a_{i_k} - l_{i_k}, a_{i_k} + u_{i_k})$. In order to achieve this goal, some other cells of the contingency table may need to be suppressed also. Thus the aim of the suppression is to find a set $C = \{j_1, \ldots, j_c\}$ that is as "small" as possible and that, for each $k = 1, \ldots, p$, there are two feasible contingency tables y and z with the following properties:

1. $y_{i_k} \geq a_{i_k} + u_{i_k}$,
2. $z_{i_k} \leq a_{i_k} - l_{i_k}$,
3. $y_i = z_i = a_i$ for $i \notin P \cup C$.

In some cases, only the bound u_{i_k} is given. Then we do not need to meet the requirement for the existence of z.

A.1. Exact Solution

We use a binary variable array $x = (x_1, \cdots, x_n)$ to represent whether each cell is suppressed. That is,

$$x_i = \begin{cases} 1 & \text{if } i \in P \cup C \\ 0 & \text{otherwise} \end{cases}$$

Let c be an array of subjective weight on cells and y^k, z^k be variables representing the potential feasible contingency tables for the condition $i_k \in P$. Then the suppression problem could be formulated as the following questions.

$$\text{minimize } c^T x \tag{4}$$

subject to

$$\begin{cases} a_i - a_i x_i \leq y_i^k \leq a_i + x_i T \text{ for } a_i \neq * \\ a_i - a_i x_i \leq z_i^k \leq a_i + x_i T \text{ for } a_i \neq * \\ y_{i_k}^k \geq a_{i_k} + u_{i_k}, k = 1, \ldots, p \\ z_{i_k}^k \leq a_{i_k} - l_{i_k}, k = 1, \ldots, p \\ Hy^k = 0; Hz^k = 0, k = 1, \ldots, p \\ x_i = 1 \text{ for } i \in P \\ x \in \{0,1\}^n; y^k \geq 0, z^k \geq 0, k = 1, \ldots, p \end{cases} \tag{5}$$

where T is a large enough integer. The above mixed integer programming (MIP) could be split into small MIPs. That is, for each $i_k \in P$, one can construct an MIP and solve it.

A.2. Heuristic methods

When the number of variables increase, it may be infeasible to solve the MIPs in the previous section. Fagan [9] recommended several heuristic mehtods. In the following, we describe one that is useful for our problem.

Let y and z be n-ary variables, and c be the subject value such that it takes 0's on known parts of $P \cup C$. We hope that $a + y - z$ will be the feasible contingency table witnessing the fact that the public cannot infer the upper bound of a_{i_k} within an error of u_{i_k} if we suppress these cells j with $y_j \neq z_j$. That is, we want to have $a_{i_k} + y_{i_k} - z_{i_k} = a_{i_k} + u_{i_k}$ and keep the nonzero entries in $y - z$ as small as possible according to the subjective weight. The heuristic formulation is as follows:

$$\text{minimize } c^T(y + z) \tag{6}$$

subject to

$$\begin{cases} z_i \leq a_i \text{ for } a_i \neq * \\ H(y - z) = 0 \\ y_{i_k} - z_{i_k} \geq u_{i_k} \\ y \geq 0, z \geq 0 \end{cases} \tag{7}$$

For the lower bound, we have the similar linear programming formulations as follows.

$$\text{minimize } c^T(y + z) \tag{8}$$

subject to

$$\begin{cases} z \leq a \\ H(y - z) = 0 \\ z_{i_k} - y_{i_k} \geq l_{i_k} \\ y \geq 0, z \geq 0 \end{cases} \tag{9}$$

Remark. In some cases, it may be possible that the bounds l_{i_k} and u_{i_k} are not given directly. That is, instead of giving l_{i_k} and u_{i_k}, only $L_{i_k} = a_{i_k} - l_{i_k}$ and $U_{i_k} = a_{i_k} + u_{i_k}$ are given. When a_{i_k} is given, then one can easily compute l_{i_k} and u_{i_k} directly. If a_{i_k} iis unknown, then one may estimate the values of l_{i_k} and u_{i_k} roughly as

$$l_{i_k} = u_{i_k} = \frac{U_{i_k} - L_{i_k}}{2}.$$