NISS

Software Systems for Tabular Data Releases

Alan F. Karr, Adrian Dobra, Ashish P. Sanil,
and Stephen E. Fienberg

Technical Report Number 125
March, 2002

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

Software Systems for Tabular Data Releases

Adrian Dobra, Alan F. Karr, Ashish P. Sanil
National Institute of Statistical Sciences, Research Triangle Park, NC 27709-4006, USA
{adobra,karr,ashish}@niss.org
Stephen E. Fienberg
Carnegie Mellon University, Pittsburgh, PA 15213 USA
fienberg@stat.cmu.edu

Abstract

We describe two classes of software systems that release tabular summaries of an underlying database. *Table servers* respond to user queries for (marginal) sub-tables of the "full" table summarizing the entire database, and are characterized by dynamic assessment of disclosure risk, in light of previously answered queries. *Optimal tabular releases* are static releases of sets of sub-tables that are characterized by maximizing the amount of information released, as given by a measure of data utility, subject to a constraint on disclosure risk. Underlying abstractions — primarily associated with the query space, as well as released and unreleasable sub-tables and frontiers, computational algorithms and issues, especially scalability, and prototype software implementations are discussed.

1 Introduction

Federal statistical agencies, and many other organizations as well, must balance concern over confidentiality of their data — in particular, identities of data subjects and sensitive attributes — with their obligation to report information to the public [10]. Advances in information technology threaten confidentiality: for example, powerful capabilities enable record linkage across multiple databases. However, new technologies can also protect confidentiality while meeting user needs in innovative ways.

In this paper we describe two classes of software systems developed by the National Institute of Statistical Sciences (NISS) under its Digital Government (DG) project [29]:

Table Servers disseminate tabular summaries of statistical data in response to user queries for marginal sub-tables of a large (e.g., 40 dimensions with 4 categories each) contingency table containing counts or sums. The distinctive characteristic of table servers is that they *evaluate disclosure risk dynamically*, in light of previously answered queries.

Optimal Tabular Release technologies calculate (for subsequent release) fixed sets of marginal sub-tables of a single large table that maximize the information released, subject to a constraint on disclosure risk. They are distinguished by accounting explicitly for both the value and the risk of the information that is disseminated.

The central challenge to building both classes of systems are development of the underlying abstractions, discussed in §3, and construction of scalable algorithms that implement the abstractions, which are described in §4 and §5.

2 Background

Here we introduce basic concepts associated with evaluation and reduction of disclosure risk for tabular data releases.

Consider a database $\mathcal{D}$ of microdata elements D_i, each consisting of n_N numerical attributes $(\mathrm{NA}_{i1}, \ldots, \mathrm{NA}_{in_N})$ and n_C categorical attributes $(\mathrm{CA}_{i1}, \ldots, \mathrm{CA}_{in_C})$. Some of the latter may be derived from numerical attributes by quantization ("binning").

Tabular Data Summaries. In this paper we focus on *count (frequency) tables* constructed from $\mathcal{D}$; these are cross-tabulations of $\mathcal{D}$ indexed by subsets $\mathcal{C}$ of the categorical attributes. If $\mathcal{C}$ contains attributes $c_1, \ldots, c_k$ — k is termed the *dimension* — then the count table $\mathrm{CT}(\mathcal{C})$ is a k-dimensional array counting how many microdata elements have each combination of category values associated with $\mathcal{C}$. Specifically,

$$\mathrm{CT}(\mathcal{C})(h_1, \ldots, h_k) = \#\{i : \mathrm{CA}_{ic_1} = h_1, \ldots, \mathrm{CA}_{ic_k} = h_k\}, \tag{1}$$

where each h_j is a possible category value for categorical attribute c_j. There are 2^{n_C} count tables constructible from D, ranging from the case $\mathcal{C} = \emptyset$ (which by convention contains the grand total of elements in $\mathcal{D}$) to the case that $\mathcal{C}$ contains all n_C of the categorical attributes — the "full table." Tables other than the full table are termed *sub-tables*.

(The other principal form of tabular summaries — *magnitude tables* — are defined by a subset of the categorical attributes and a single numerical attribute. Each cell, rather than containing a count of data elements with specified category values, contains the *sum* of the specified numerical attribute over all such data elements.)

Risk Evaluation. The simplest method for evaluation of disclosure risk in count tables representing an entire population is the n–rule [43]: the sub-table $\mathrm{CT}(\mathcal{C})$ is too risky if any non-zero cell count is less than n, and safe to release otherwise. In practice, n is often 3. Analogously, for magnitude tables there is the (n, p)–rule: a sub-table is too risky if the underlying cell count is less than n *or* if any of the data elements contributing to the sum in that cell dominates in the sense that it comprises more than p% of the sum. A typical value in practice is $p = 60\%$.

For tables with sample counts, uniqueness in the sample may or may not pose a risk of disclosure. Thus the literature [17, 18, 32] looks at the proportion of sample uniques that are in fact population uniques. If this is small, then the risk is small even in the presence of a large number of counts of 1 or 2 in the table. A major determinant of this proportion is the sampling fraction.

For count tables there are statistical and other methodologies to compute bounds on entries in the full table in terms of released information such as marginal cross-tabulations, and the tightness of such bounds forms an alternative measure of risk, which we employ in §3 and 5. In the important cases that the released marginals constitute a decomposable model, the bounds are both sharp and computable using scalable methods [4, 9]; see §3.2.

Risk Reduction. Numerous strategies exist for reducing disclosure risk and thereby protecting against identity disclosure for subjects of tabular data [42]. Some of these, such as aggregation [21, 22, 25], cell suppression [6, 31], perturbation [11, 12, 19, 20] and controlled rounding [5], operate on the full table itself. Other methods, such as data swapping [7] and jittering (addition

of noise to numerical data attributes) [43], operate directly on the underlying database, prior to formation of tables.

The software systems described in this paper, in effect, control disclosure risk by not responding to user queries that are "too risky." Table servers (§3.1 and 4) use dynamic evaluation of the risk of queries, in light of previously released information. The technology for optimal tabular releases described in §3.2 and 5 constructs *sets* of sub-tables that maximize the information to be disseminated but prevent identify disclosure nevertheless, by satisfying a constraint on disclosure risk. Both kinds of systems share two important characteristics.

First, disclosure risk depends on the *entire collective of released information*, whether dynamically for table servers or statically for optimal tabular releases. By contrast, some extant systems evaluate risk only for queries in isolation, ignoring the risk associated with interacting queries.

Second, confidentiality is ensured by *restricting the level of detail at which data are released*, rather than — as is the case for data swapping and jittering — distorting the data. Heuristically, table servers and optimal tabular releases always tell the truth, albeit not the whole truth. This is especially important for statistical inference from the data: conclusions may be more uncertain, but will not be erroneous.

3 Abstractions

In this section, we discuss the abstractions underlying table servers (§3.1) and optimal tabular releases (§3.2).

3.1 Table Servers

Queries and Responses. User queries to a table server are for (marginal) sub-tables of a large count table: the user forms the query by specifying the variables to be tabulated (see Figure 3). Potential responses include the requested sub-table (as text, visualized or in XML), its "projection" onto the released frontier (see below) and risk-reduced modifications of the requested sub-table. The query may also simply be refused, although this alone may be informative.

Query Space. Simplifying the notation from §2, let $\mathbf{T}$ be a K-way count table, is the "underlying table." Let $[v_{i_1} v_{i_2} \ldots v_{i_j}]$ denote the marginal sub-table defined by variables $v_{i_1}, v_{i_2}, \ldots, v_{i_j}$.

The query space $\mathcal{Q}$ for a table server based on $\mathbf{T}$ contains all 2^K sub-tables of and is partially ordered by set inclusion of variables in sub-tables. Denoting the partial order by "Child" $\prec$ "Parent," then for example, in the demographic data set underlying the prototype in Figure 1,

$$\Big[\mathsf{Age, Education, Race}\Big] \prec \Big[\mathsf{Age, Education, EmployerType, Race, SalaryLevel}\Big].$$

Released Set and Frontier. At any time t, the set $\mathcal{R}(t)$ of all sub-tables released through time t contains both: *direct releases* in response to queries and *indirect releases* — previously unreleased children of direct releases. This set is specified completely by the *released frontier*

$\mathcal{RF}(t)$ consisting of the maximal elements of $\mathcal{R}(t)$ — those with no released parents. That is, $T \in \mathcal{R}(t)$ lies on the frontier $\mathcal{RF}(t)$ if and only if there is no $T' \in \mathcal{R}(t)$ such that $T \prec T'$. In Figure 1, which shows a prototype table server for an 8-variable demographic database, $\mathcal{R}(t)$ lies in the lower left portion of the query space, and $\mathcal{RF}(t)$ is its upper boundary.

Risk Criterion. Underlying release decisions is a risk criterion **RC** defined on subsets of $\mathcal{Q}$. The conceptual basis of the system is that the *collective* risk of the released information cannot exceed a threshold: at all times the system must satisfy

$$\mathbf{RC}(\mathcal{R}(t)) \leq \alpha,$$

where α is a risk threshold set by the operators. To emphasize the dynamic nature of table servers, no requested sub-table T can be released at t that — *together with previously released information* — would cause the threshold to be exceeded:

$$\mathbf{RC}(\mathcal{R}(t) \cup T) > \alpha. \tag{2}$$

Note that this formulation assumes (conservatively but sensibly) that all users can communicate with one another.

Reflecting historical usage, a typical risk criterion is accuracy of bounds based on $\mathcal{R}(t)$ for sensitive (small count) cells in the full table. Such bounds can be computed using network methods [6, 30] and NISS-developed generalizations of the "shuttle algorithm" [4]. There are also exact techniques for special cases, which are described in detail in §5.

Unreleasable Set and Frontier. Whenever an answered query releases previously unreleased information, other queries become unanswerable as a result. Consequently, at time t there are an *unreleasable set* $\mathcal{U}(t)$ of sub-tables T whose release is too risky, in the sense that (2) would hold. The set $\mathcal{UF}(t)$ of minimal elements of $\mathcal{U}(t)$ is known as the *unreleasable frontier*. In Figure 1, the unreleasable set lies to the upper right; the unreleasable frontier is its lower boundary.

Release Rules. When a query requesting release of a sub-table T is received at time t, the system must decide how to respond. If $T \in \mathcal{R}(t)$, the response is clear: the requested information has been released previously, so the query simply answered again. However, when the request is for an unreleased sub-table, a *release rule* must be invoked to determine whether it will be answered.

The simplest rule is the *myopic rule* of releasing any requested sub-table that is not too risky: T will be released at t as long as

$$\mathbf{RC}(\mathcal{R}(t) \cup T) \leq \alpha.$$

The myopic rule, however, allows the table server to take very large steps, by releasing sub-tables containing many more variables than those that have been released previously. To prevent this, the table server can allow only sub-tables adding but one variable to a previously released sub-table — that is, those with a first generation child on the released frontier $\mathcal{RF}(t)$ — to be eligible for release.

More subtle issues arise, though. Typically, it is necessary to prevent a single user (or a set of colluding users) from driving the table server into a region of $\mathcal{Q}$ that suits their needs but not those of other users. One defense against this is to use release rules that are biased against releases that

Figure 1: Java table server prototype. Levels in the visualization of the query space correspond to sub-table dimension, with the full table **T** at the top and the 0-dimensional sub-table ar the bottom. Also shown is the effect of releasing the 5–way sub-table indicated by the cursor on $\mathcal{R}(t)$ and $\mathcal{U}(t)$, as well as on the associated frontiers $\mathcal{RF}(t)$ and $\mathcal{UF}(t)$.

cause many other sub-tables to become unreleasable. Thus, releasing T at t would, all other things being equal, be deemed undesirable if it were to lead to a large increase in the size of $\mathcal{U}(t)$.

Another alternative is release rules that incorporate a suitably defined *value* of releasing T [13, 23, 37]. One example of value is the accuracy with which the full table **T** can reconstructed from $\mathcal{R}(t) \cup T$ by means of iterative proportional fitting [1].

The decision whether to respond to a query need not be taken immediately. Instead, for example, the system could accumulate requests for different sub-tables and employ user interest as a measure of value.

While intriguing, table servers pose significant issues of scale, of user equity (to prevent the system from being driven in a direction that serves only a small set of users), and of the scientific, statistical and policy bases for release rules. These and other considerations led to development of the optimal tabular release technology described next.

3.2 Optimal Tabular Releases

Optimization Formulation. An optimal tabular release (OTR) is constructed by solving an optimization problem of the form

$$\begin{aligned} &\max_{\mathcal{R}\subseteq\mathcal{Q}} \mathbf{DU}(\mathcal{R}) \\ &\text{s.t. } \mathbf{RC}(\mathcal{R}) \leq \alpha, \end{aligned} \tag{3}$$

where **DU** is a measure of the utility of the released data, **RC** is a risk criterion (examples of both appear below) and α is a risk threshold. That is, the utility of the released information — the set $\mathcal{R}$ of sub-tables — is maximized, subject to an upper bound on the disclosure risk. This risk–utility approach builds on other risk–utility formulations [13, 23, 38, 39] being investigated under the NISS DG project.

Risk Criteria and Thresholds. A typical risk criterion is tightness of bounds based on $\mathcal{R}$ for small count cells in the full table; a specific example is

$$\mathbf{RC}(\mathcal{R}) = -\min\Big\{\mathrm{UB}(C,\mathcal{R}) - \mathrm{LB}(C,\mathcal{R}) : 0 < \#\{C\} \leq 3\Big\}, \tag{4}$$

where C is a generic cell in the full table and $\#\{C\}$ is its count value. In (4), $\mathrm{UB}(C,\mathcal{R})$ and $\mathrm{LB}(C,\mathcal{R})$ are upper and lower bounds on $\#\{C\}$ determined from $\mathcal{R}$. Techniques to compute such bounds are noted in §3.1. Other methods are available for special cases: see §5.

Data Utility. An illustrative measure of data utility is

$$\mathbf{DU}(\mathcal{R}) = \#\{\mathcal{R}\}, \tag{5}$$

the number of sub-tables contained in $\mathcal{R}$. Alternatives include the number of cells in these sub-tables and the number of degrees of freedom represented by them.

As noted in §3.1, another measure of utility is a global measure of the accuracy — measured, for example, by the average per cell mean squared error — with which the full table **T** can be reconstructed from $\mathcal{R}$ by means of iterative proportional fitting [1].

4 Table Servers: System Design and Prototypes

A prototype table server, implemented as a Java application, is shown in Figure 1. This prototype is valuable for its engaging, but non-scalable, visualization of the query space. It operates on an 8–dimensional full table of data from the 1993 Current Population Survey (CPS) [2, 3]. The underlying data come from a sample survey carried out by the US Census Bureau, which monthly gathers data on approximately 50,000 households across the US.

As seen in the figure, the critical sub-tables at any given time as well as consequences of releasing particular sub-tables can be readily discerned.

Figure 2 shows the architecture of a more powerful table server implemented using server-side Java [33]. This prototype operates on a 14-dimensional full table derived from the 1994 and 1995 CPSs. The full table has approximately 435,000,000 cells, and is extremely (but realistically!) sparse, principally as a result of the small sample size.

Figure 3 shows the user-input screen. If the requested sub-table lies on or below $\mathcal{RF}(t)$, it is provided immediately (via a screen display if the sub-table is "small," and otherwise via XML download). Releases are governed by the myopic rules that consider only those sub-tables that are "at most one step" away from $\mathcal{R}(t)$ as candidates for release. Requests for sub-tables that lie one step or higher than $\mathcal{R}(t)$ are immediately denied. Disclosure risk for viable sub-tables is evaluated in real time, and the sub-table is released if it is not considered too risky. The query history database, with tables for users, queries and the time trajectories of $\mathcal{RF}(t)$ and $\mathcal{UF}(t)$, is maintained in a MySQL database server [44]. A frontier display facility, shown in Figure 4, monitors evolution of $\mathcal{RF}(t)$.

As shown in Figure 2, the HTML-based user-query processing as well as overall program logic is implemented by Java Servlets [34]; we use Apache's Tomcat [36] as the servlet engine. Beyond straightforward tasks such as interaction with the database and output generation, the most significant work performed is the real-time computation of disclosure risk.

For maximum efficiency, the computations are done by a native C-program. Because standard implementations of risk computation procedures based on cell-bounds do not scale to tables with over 400 million cells, we do the risk calculation in real time using a generalized version of the heuristic "Shuttle Algorithm" [4] along with specially crafted data structures. The risk computation procedure employs data structures and algorithms that exploit sparsity of the full table and the fact that $\mathcal{R}(t)$ and $\mathcal{U}(t)$ are characterized completely by $\mathcal{RF}(t)$ and $\mathcal{UF}(t)$. The data structures represent the full table as a hash table in such a way that the iterations involved in the algorithm can be organized so as to minimize the inner-loop computations. This results in a speedup significant enough to permit us to do the calculation in real time!

5 Optimal Tabular Releases

There are severe scalability issues for the basic OTR formulation in (3) with respect to the number dimensions of the full table. If the full table contains n_C attributes, then there are on the order of $2^{2^{n_C}}$ choices of $\mathcal{R}$ in (3), which for even moderate values of n_C (e.g, 20) is simply too many, especially if either $\mathbf{DU}(\cdot)$ or $\mathbf{RC}(\cdot)$ requires intensive computation.

Here we discuss two examples of ways to circumvent this difficulty. The first restricts the released information to correspond a decomposable statistical model (§5.1), which has the effect of both decreasing the family of allowable releases $\mathcal{R}$ in (3) and simplifying dramatically computation of the risk. Even so, computational demands are formidable: because sets of marginal totals correspond to the minimal sufficient statistics of log-linear models fitted to the full table [1], searching for OTRs has many of the same characteristics of searching through all possible log-linear models or the subclass of all decomposable models [14, 27, 40].

The second way (§5.2) "solves" (3) heuristically, by ordering the marginal sub-tables of $\mathbf{T}$ according to a particular notion of risk, and then constructing the heuristically optimal release $\mathcal{R}^*$ by greedy method adding increasingly risky sub-tables until no more can be added without violating a global risk threshold of the form (4).

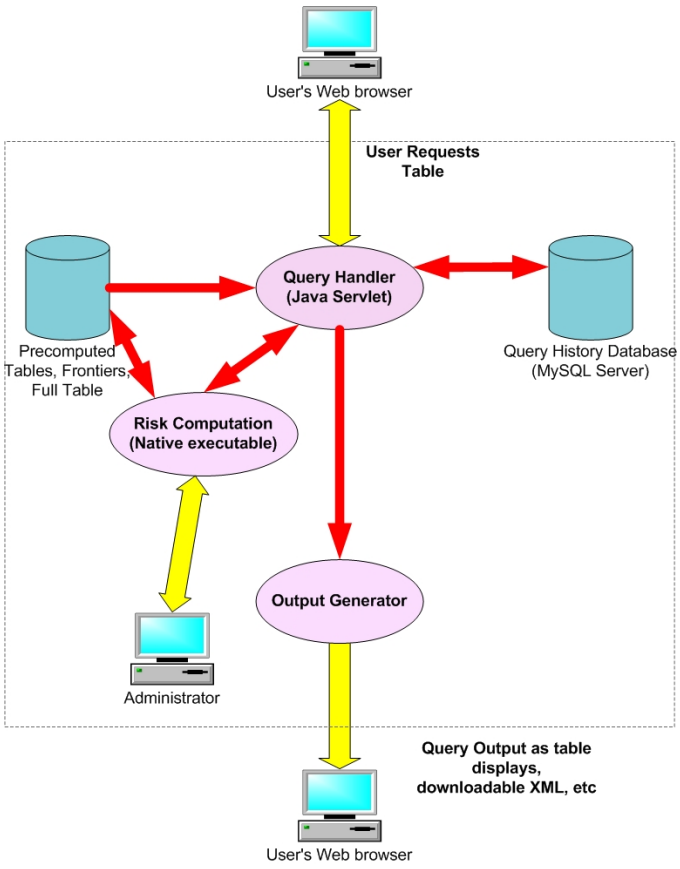

Figure 2: Table server prototype: System architecture. The system can handle 14-dimensional (and higher!) tables. User interaction occurs through a standard Web browser. Output formats include screen display and XML. See §4 for details.

5.1 Decomposable Releases

By restricting $\mathcal{R}$ in (3) to have the "right" special structure, we can overcome both scalability and other computational challenges. We exploit, in this context, the statistical theory on graphical models [24, 28, 41], which shows that the conditional dependencies induced by the sub-tables in $\mathcal{RF}$ among the variables cross-classified in a table of counts consistent with $\mathcal{RF}$ can be visualized by means of an independence graph. Each vertex in this graph represents a variable in the underlying table. We draw an edge between two vertices if and only if the two-dimensional array defined by the variables associated with these vertices belongs to $\mathcal{R}$.

Decomposable graphs [24] are the special class of graphs that can be "broken" into components such that (i) every component is associated with exactly one fixed sub-table in the frontier; and (ii) no released sub-table is "split" between two components. Reducible graphs [35, 26] are generalizations of decomposable graphs. A reducible graph is one that can be at least partially decomposed, although the resulting components of the decomposition may correspond to more than

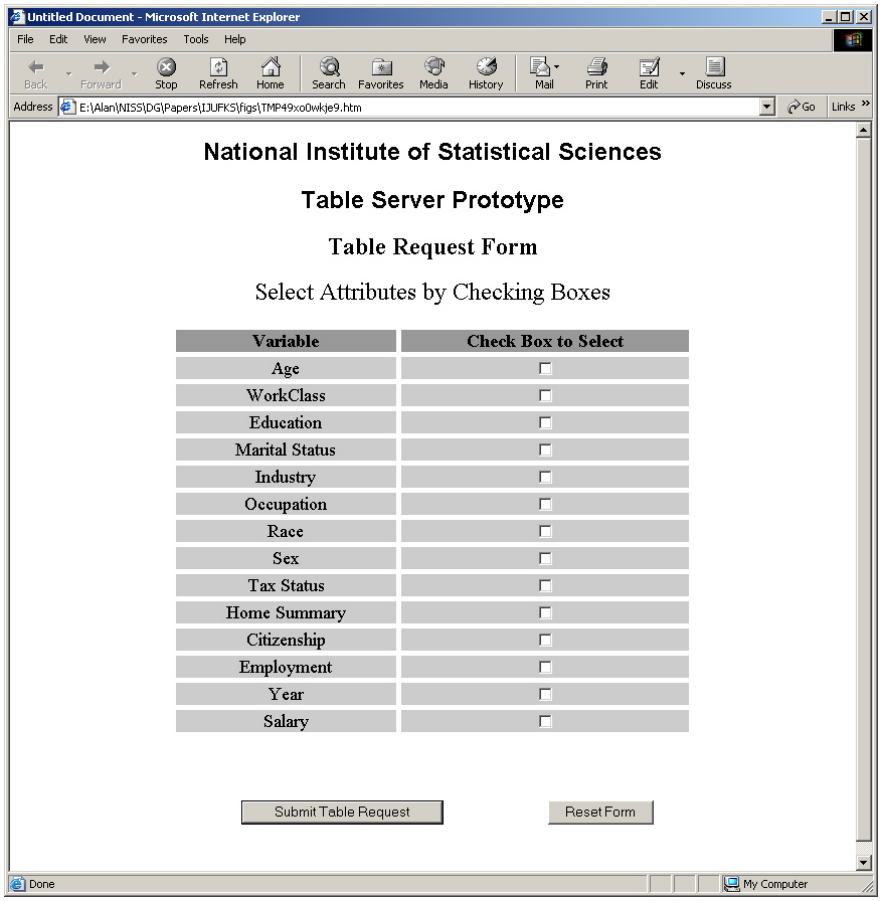

Figure 3: Table server prototype: User input screen. Queries are posed by selecting the variables in the desired sub-table.

one fixed sub-table.

If the independence graph of a graphical model is decomposable (reducible), then the model is said to be decomposable (reducible). In a similar way, if every sub-table in $\mathcal{RF}$ defines a component of the independence graph, the set of released sub-tables $\mathcal{R}$ is called *decomposable*– see Figure 5. In this case [9], UB(C, $\mathcal{R}$) and LB(C, $\mathcal{R}$) in (4) can be expressed as explicit functions of the cell counts in sub-tables in $\mathcal{R}$.

Problem Formulation. Specifically, if we require that the sub-tables in $\mathcal{R}$ constitute the minimal sufficient statistics of a decomposable graphical model [1, 14, 24] (see Figure 5), then first of all the number of candidate releases decreases dramatically. But, more important, in this case, both UB(C, $\mathcal{R}$) and LB(C, $\mathcal{R}$) in (4) can be expressed as *explicit* functions of $\mathcal{R}$ [9].

For simplicity we adopt the data utility measure of (5) — the number of sub-tables released.

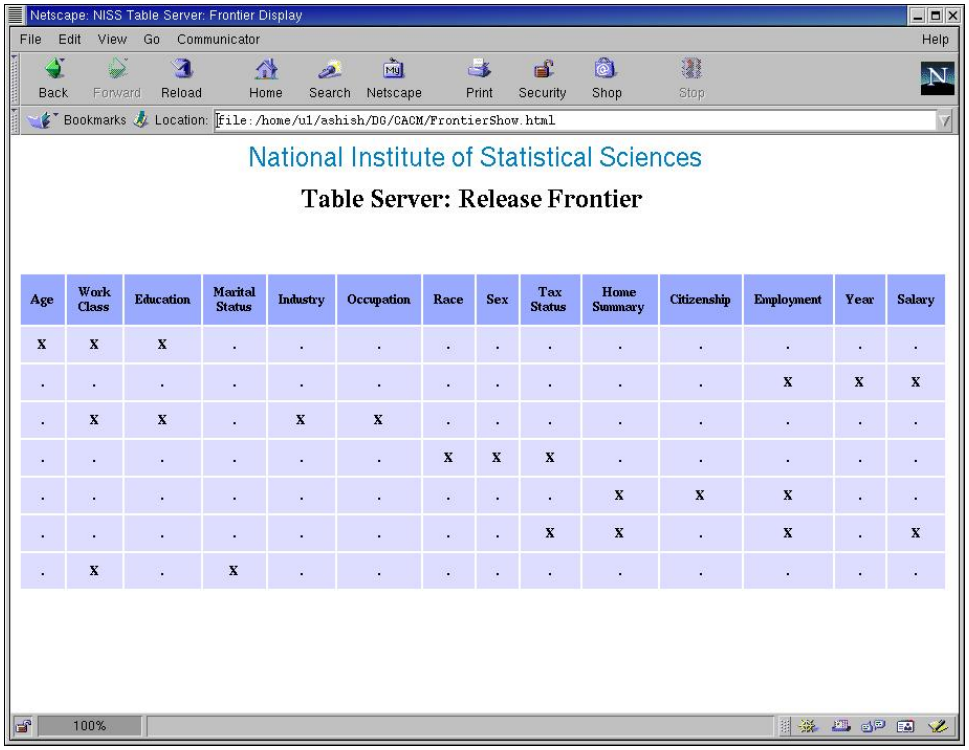

Figure 4: Table server prototype: Released frontier display. The display lists the sub-tables comprising $\mathcal{RF}(t)$.

Therefore, in this case (3) becomes

$$\begin{aligned} &\max_{\mathcal{R}\subseteq\mathcal{Q}} \#\{\mathcal{R}\} \\ &\text{s.t.} \quad \min\left\{\mathrm{UB}(C,\mathcal{R}) - \mathrm{LB}(C,\mathcal{R}) : 0 < \#\{C\} \leq 3\right\} \geq \beta \\ &\quad\ \mathcal{R} \text{ is decomposable.} \end{aligned} \tag{6}$$

The choice of 3 as bound width in (6) is illustrative. By "$\mathcal{R}$ is decomposable" we mean that the released frontier comprises the minimal sufficient statistics of a decomposable model.

More generally, when the released marginals are associated with a *reducible* graphical model [24], the computational effort associated with (6) can be decreased significantly by "divide-and-conquer" techniques [9].

Computational Implementation. Even (6) cannot be solved by enumeration: the number of decomposable models grows exponentially with the number of variables in **T**. One way to cope with this problem is to employ simulated annealing — an iterative Monte Carlo approach for computing a local maximum of $\#\{\cdot\}$ by generating a random sample from the distribution

$$\pi(\mathcal{R}) \propto \exp\left(\frac{\#\{\mathcal{R}\}}{T}\right), \tag{7}$$

where T is a scale parameter interpretable as *temperature*, which is decreased toward 0 as the algorithm progresses. Given a current state $\mathcal{R}_j$, a new decomposable model $\mathcal{R}_{j+1}$ is selected from a uniform distribution on a neighborhood $N(\mathcal{R}_j)$ of $\mathcal{R}_j$. If $\#\{\mathcal{R}_{j+1}\} \geq \#\{\mathcal{R}_j\}$ (i.e., data utility increases), then $\mathcal{R}_{j+1}$ becomes the current state with probability 1. Otherwise, if $\#\{\mathcal{R}_{j+1}\} < \#\{\mathcal{R}_j\}$, $\mathcal{R}_{j+1}$ is accepted with probability

$$\min\left\{\exp\left(\frac{\#\{\mathcal{R}_{j+1}\} - \#\{\mathcal{R}_j\}}{T}\right), 1\right\}. \tag{8}$$

At higher temperatures, the Markov chain $(\mathcal{R}_j)$ can "escape" local optima of (6), while as T approaches 0, its movements will concentrate in a smaller and smaller region around (what is hoped to be) the global maximum.

In our implementation, the neighborhood $N(\mathcal{R})$ is taken to consist of all releases defined by decomposable independence graphs obtained by deleting or adding one edge from the graph associated with $\mathcal{R}$. Very efficient algorithms [8] exist for finding $N(\mathcal{R})$. Because any two decomposable graphs can be linked by a sequence of decomposable graphs that differ by exactly one edge [24], the resulting Markov chain is irreducible, as required for simulated annealing to work.

Example. The simulated annealing algorithm used to solve (6) was run on a 13-dimensional data set extracted from the 1994 and 1995 CPSs, which contains 299,285 microdata records. (These data were derived from the 14-dimensional data set underlying the table server prototype of §4.) The use of CPS data is illustrative: because the data are a sample, there may be no disclosure risk at all. Moreover, the records are constructed by collapsing across monthly data, but some of these may correspond to the same people. Nevertheless, for the purposes of the example, we act as if the data correspond to individuals forming a population, so that small count cells must be protected.

The variables have 5, 2, 3, 3, 5, 8, 2, 2, 3, 5, 2, 3 and 2 categories, respectively. The full table is quite sparse: only 41,672 of the 2,592,000 cells contain non-zero entries. Of these, 22,996 cells contain a count of "1", 6,435 cells contain a count of "2", and 3,032 have a count of "3." Therefore, almost 80% of the non-zero cells in the full table are subject to disclosure risk [43].

The simulated annealing algorithm was started with $\mathcal{R}_0$ containing the thirteen one-way sub-tables, which is "safe." The algorithm converged to a set of releasable sub-tables $\mathcal{R}^*(\beta = 3)$ whose frontier is

$$\begin{aligned}\mathcal{RF}^*(\beta = 3) \;=\; \Big\{&[1, 7, 8, 11, 12, 13], [7, 8, 11, 12, 13], [2, 3, 7, 8, 11, 12, 13],\\ &[2, 3, 4, 7, 8, 11, 13], [2, 5, 7, 8, 11, 13], [5, 6, 7, 8, 11, 13],\\ &[2, 4, 7, 9, 11, 13], [2, 7, 8, 10, 11, 13]\Big\},\end{aligned}$$

and which contains five 6-way and two 7-way sub-tables. The release contains a total of 351 sub-tables, representing 4.25% of the 8,191 sub-tables of the full 13-way table. See Table 1 for details.

It is interesting to contrast $\mathcal{R}^*(\beta = 3)$ with the set of released sub-tables whose frontier consists of *all* 3-way sub-tables. (Such releases are common practice in the statistical agencies [16].)

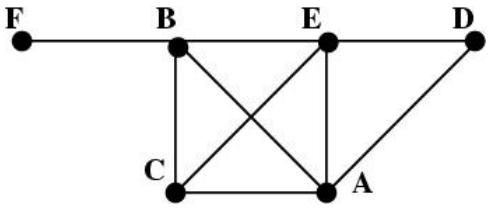

Figure 5: Independence graph associated with three marginal sub-tables [BF], [ABCE] and [ADE] of a six-way table [ABCDEF], visualizing the dependency patterns induced by the released marginals. The graphical model with minimal sufficient statistics [BF], [ABCE] and [ADE] is decomposable [24].

This set contains 26 sub-tables more than $\mathcal{R}^*(\beta = 3)$, but arguably $\mathcal{R}^*(\beta = 3)$ provides users with more information because 186 (or 52%) sub-tables in $\mathcal{R}^*(\beta = 3)$ have dimension four or greater. In addition, because the model whose frontier consist of all 3-dimensional sub-tables is not decomposable, determining whether it is releasable constitutes a formidable computational burden.

Larger values of the risk threshold β decrease the information that can be released. For $\beta = 4$, the optimal released frontier is

$$\begin{aligned}\mathcal{RF}^*(\beta = 4) \;=\; \Big\{&[1, 2, 8, 11, 13], [2, 3, 4, 7, 8, 11, 13], [2, 3, 7, 8, 11, 12, 13],\\ &[3, 4, 5, 7, 8, 13], [3, 7, 8, 9, 11, 13], [2, 7, 8, 10, 11, 13], [3, 6, 8, 12, 13]\Big\},\end{aligned}$$

and 319 sub-tables (3.89%) are released. For $\beta = 5$, the algorithm produced

$$\begin{aligned}\mathcal{RF}^*(\beta = 5) \;=\; \Big\{&[1, 2, 7, 9, 13], [2, 7, 8, 9, 11, 13], [2, 3, 7, 8, 11, 13],\\ &[2, 3, 4, 7, 11, 13], [5, 7, 8, 11, 12, 13], [3, 6, 8, 11, 13], [7, 8, 10, 11, 12]\Big\},\end{aligned}$$

releasing 239 sub-tables.

5.2 Heuristically Ordered Marginals

Two potential objections to the approach in §5.1 are that even with the restriction to decomposable models the computational demands are formidable, and that the decomposability restriction itself is not justifiable. We recognize these, but are not in complete sympathy with either. In any case, however, it does make sense to consider other approaches to the OTR problem. As in §5.1, we again act as if the example data represent a population of individuals.

Approach. The heuristic method discussed here is straightforward: we order the elements T of the query space $\mathcal{Q}$ in terms of a particular measure of risk, and add them to the release in order of increasing risk until no more can be added without exceeding a risk threshold. The details of this greedy procedure are somewhat complex, however, and we lay them out step-by-step.

Dimension	Number of Released Sub-tables	Total Number of Sub-tables	Percent
1	13	13	100%
2	52	78	66.67%
3	100	286	34.97%
4	105	715	14.69%
5	61	1,287	4.74%
6	18	1,716	1.05%
7	2	1,716	0.12%

Table 1: Breakdown of the released set of sub-tables $\mathcal{R}(\beta = 3)$. The columns show the dimension of sub-tables, how many sub-tables of that dimension are in $\mathcal{R}(\beta = 3)$, the total number of sub-tables and the percentage of released sub-tables.

First, we term a cell C in the full table $\mathbf{T}$ *at risk* if $0 < \#\{C\} \leq 2$ (i.e., if its count is 1 or 2). These are the cells that must be protected.

Second, by analogy to (4), given a *bound width* w, we term a release $\mathcal{R}$ *bad at width* w if

$$\min\left\{\text{UB}(C, \mathcal{R}) - \text{LB}(C, \mathcal{R}) : C \text{ is at risk}\right\} \leq w. \tag{9}$$

That is, $\mathcal{R}$ is bad at w if at least one at risk cell can be bounded within w on the basis of the sub-tables contained in $\mathcal{R}$.

Third, the *most parsimonious* release $\text{MPR}(T)$ associated with a sub-table $T \in \mathcal{Q}$ is obtained by releasing T together with the one-dimensional sub-tables corresponding to all variables that do not appear in T. For example, if $T = [1, 2, 3]$ for a six-dimensional table then $\text{MPR}(T) = \{[1, 2, 3], [4], [5], [6]\}$.

In order to assess how "dangerous" a sub-table T would be if it were released, it suffices to consider $\text{MPR}(T)$, since it is embedded in any other possible release containing T. Moreover, because it is most parsimonious, $\text{MPR}(T)$ has the loosest bounds of all such releases. Finally, $\text{MPR}(T)$ is decomposable, so that these bounds can be computed explicitly.

Next, a sub-table T is then termed *bad at width* w if $\text{MPR}(T)$ is bad at w in the sense of (9).

Fourth, on the basis of the computations outlined, we can define for each $T \in \mathcal{Q}$ a *critical width*

$$w^*(T) = \max\{w : T \text{ is bad at } w\}. \tag{10}$$

The higher $w^*(T)$, the safer it is to release T. Consequently, a sub-table T_1 is said to be *more dangerous* than another sub-table T_2 if $w^*(T_1) < w^*(T_2)$.

To construct a release, we place the sub-tables in a list $\mathcal{L}$ in decreasing order with respect to their critical widths, that is, from least to most dangerous. Tables with the same critical width are put in decreasing order with respect to their dimension, since to maximize the amount of released information, we would prefer to release a higher-dimensional sub-table.

Let $T_1, T_2, \ldots, T_L$ be the sub-tables in the order in which they appear in $\mathcal{L}$. We want to identify the unique rank $l_0 \in \{1, 2, \ldots, L\}$ such that $\mathcal{R}^* = \{T_1, \ldots, T_{l_0}\}$ is releasable but $\mathcal{R}' =$

				B	no		yes	
F	E	D	C	A	no	yes	no	yes
neg	< 3	< 140	no		44	40	112	67
			yes		129	145	12	23
		≥ 140	no		35	12	80	33
			yes		109	67	7	9
	≥ 3	< 140	no		23	32	70	66
			yes		50	80	7	13
		≥ 140	no		24	25	73	57
			yes		51	63	7	16
pos	< 3	< 140	no		5	7	21	9
			yes		9	17	$\boxed{1}$	4
		≥ 140	no		4	3	11	8
			yes		14	17	5	$\boxed{2}$
	≥ 3	< 140	no		7	3	14	14
			yes		9	16	$\boxed{2}$	3
		≥ 140	no		4	0	13	11
			yes		5	14	4	4

Table 2: Czech auto worker data from [15].

$\{T_1, \ldots, T_{l_0}, T_{l_0+1}\}$ is not. Instead of sequentially adding new sub-tables starting from the top of list $\mathcal{L}$, we employ a much more efficient bisection search strategy, as described below:

Step 0. Initialize $l_1 := 1$ and $l_2 := L$.

Step 1. Put $l_3 := \lfloor (l_1 + l_2)/2 \rfloor$.

Step 2. Check whether the set of sub-tables $\{T_1, \ldots, T_{l_3}\}$ is releasable. If so, set $l_1 := l_3$. Otherwise, set $l_2 := l_3$.

Step 4. If $l_2 = l_1 + 1$, set $l_0 := l_1$ and stop. Otherwise, go to **Step 1**.

Example. We illustrate this approach using the cross-classification of 1841 Czechoslovakian car factory workers, who took part in a prospective epidemiological study to investigate the potential risks factors for coronary thrombosis [15]. The data we examine here form the six-way table given in Table 2 where the six variables are defined as follows: A indicates whether the worker "smokes," B corresponds to "strenuous mental work," C corresponds to "strenuous physical work," D corresponds to "systolic blood pressure," E corresponds to "ratio of β and α lipoproteins," and F represents "family anamnesis of coronary heart disease."

There are three dangerous cells — one with count 1 and two with count 2. The resultant optimal release $\mathcal{R}^*$ consists of all sub-tables of $\mathbf{T} =$ [ABCDEF] other than [ACDEF] (the sub-table with

[ACDEF]	3	[ABCF]	30
[ABCDE]	5	[ABCD]	30
[ABDEF]	6	[DEF]	45
[BCDEF]	9	[ADF]	49
[ACDE]	10	[BCEF]	52
[ABCEF]	10	[CDE]	54
[ABCDF]	10	[AEF]	55
[ADEF]	12	[BDE]	56
[ABDE]	12	[ABD]	57
[BDEF]	20	[ACE]	58
[ABDF]	20	[ABF]	58
[ACDF]	21	[ACF]	59
[CDEF]	22	[ACD]	61
[BCDE]	23	[ABE]	61
[ABEF]	23	[ABC]	64
[ADE]	25	[BCDF]	68
[ACEF]	25	[DE]	119
[ABCE]	26		

Table 3: Critical widths for the most dangerous sub-tables in the example of §5.2.

the smallest critical width), [ABCDE] and [ABDEF]. Therefore $\mathcal{R}^*$ contains a total of 60 sub-tables. By contrast, the maximum number of sub-tables contained in a decomposable release is 48.

More detailed examination of the critical widths, the smallest 35 of which — corresponding to the 35 most dangerous sub-tables, are shown in Table 3), reveals intriguing structure. In particular, the sub-tables of each dimension with the smallest critical widths — [ACDEF] for dimension 5, [ACDE] for dimension 4, [ADE] for dimension 3 and [DE] for dimension 2 — form a chain with respect to the partial order $\prec$. The same was true in other examples, but whether this property holds generally — if it did, it would reveal which combinations of variables are most threatening to disclosure risk — is uncertain. A complete visualization of the critical widths appears in Figure 6.

6 Summary

The two classes of software systems we have described operate in different contexts and take quite different approaches to disclosure limitation for tabular data. Table servers are "live," responding dynamically to incoming user queries for sub-tables of the full table and assessing disclosure risk in light of previously answered queries. Table servers can be built at realistic scales, but many questions remain, especially those associated with release rules and operating policies that the user community would view as equitable.

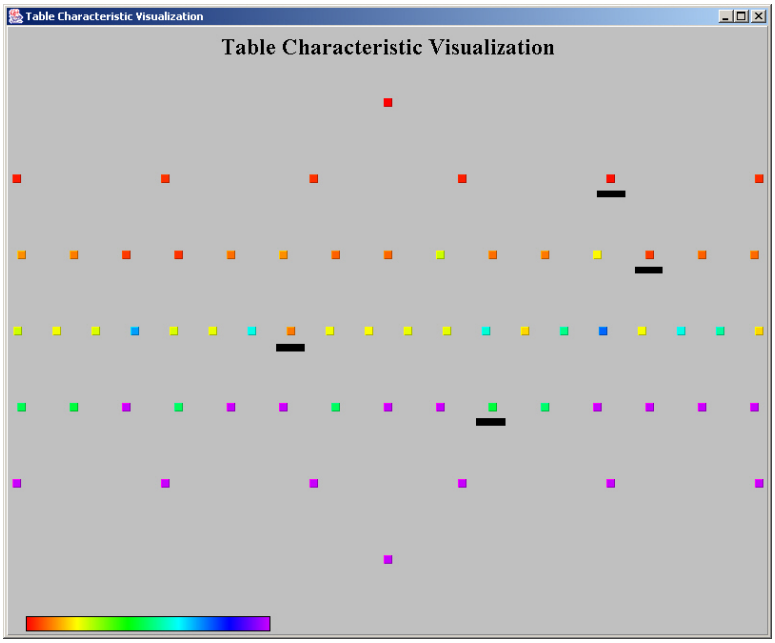

Figure 6: Visualization of the critical widths for all sub-tables in the example of §5.2. The color scale at the lower left shows values ranging from 0 (the most risky) to 255 (the least risky). The chain comprising the most dangerous table of each dimension is highlighted.

Optimal tabular releases, by contrast, are static releases of sets of sub-tables that account explicitly for both disclosure risk (as constraint) and the utility (as objective function) of the information released. Two approaches, one restricting the released frontier to correspond to a decomposable statistical model and the other a greedy algorithm that adds sub-tables in order of increasing individual risk until no more can be added without violating a global risk constraint, were described and illustrated.

Despite differences, the two classes of systems also have many similarities. As described in §3, they depend on many of the same underlying abstractions, especially released and unreleasable frontiers. Implementation in either case raises difficult issues of scalability that affect all details of algorithms from data structures to computational techniques.

Acknowledgements

The research reported here was supported in part by NSF grant EIA–9876619 to the National Institute of Statistical Sciences.

References

[1] Y. M. M. Bishop, S. E. Fienberg, and P. W. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, Cambridge, MA, 1975.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[3] US Census Bureau. Current Population Survey, 2002. Information available on-line at www.bls.census.gov/cps/cpsmain.htm.

[4] L. Buzzigoli and A. Giusti. An algorithm to calculate the lower and upper bounds of the elements of an array given its marginals. In *Proceedings of the Conference on Statistical Data Protection*, pages 131–147, Luxembourg, 1999. Eurostat.

[5] L. H. Cox. A constructive procedure for unbiased controlled rounding. *J. Amer. Statist. Assoc.*, 82:520–524, 1987.

[6] L. H. Cox. Network models for complementary cell suppression. *J. Amer. Statist. Assoc.*, 90:1453–1462, 1995.

[7] T. Dalenius and S. P. Reiss. Data swapping: A technique for disclosure control. *J. Statist. Planning Inf.*, 6:73–85, 1982.

[8] A. Deshpande, M. Garofalakis, and M. Jordan. Efficient stepwise selection in decomposable models. In *Proceedings of UAI '2001*, 2001. Available on-line at www.cs.berkeley.edu/ jordan/papers/forwardselection.pdf.gz.

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

[10] G. T. Duncan, V. A. de Wolf, T. B. Jabine, and M. L. Straf. Report of the Panel on Confidentiality and Data Access. *J. Official Statist.*, 9:271–274, 1993.

[11] G. T. Duncan and S. E. Fienberg. Obtaining information while preserving privacy: A Markov perturbation method for tabular data. In *Statistical Data Protection, Proceedings of the Conference, Lisbon*, pages 351–362, Luxembourg, 1999. Eurostat.

[12] G. T. Duncan, S. E. Fienberg, R. Krishnan, R. Padman, and S. F. Roehrig. Disclosure limitation methods and information loss for tabular data. In P. Doyle, J. Lane, J. Theeuwes, and L. Zayatz, editors, *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, pages 135–166. Elsevier, New York, 2001.

[13] G. T. Duncan and S. Keller–McNulty. Mask or impute?, 2001. Unpublished manuscript.

[14] D. Edwards. *Introduction to Graphical Modelling*. Springer–Verlag, New York, 2000.

[15] D. E. Edwards and T. Havraneek. A fast procedure for model search in multidimensional contingency tables. *Biometrika*, 72:339–351, 1985.

[16] Federal Committee on Statistical Methodology. *Report on Statistical Disclosure Limitation Methodology*. US Office of Management and Budget, Washington, 1994.

[17] S. E. Fienberg and U. E. Makov. Confidentiality, uniqueness, and disclosure limitation for categorical data. *J. Official Statist.*, 14:385–397, 1998.

[18] S. E. Fienberg and U. E. Makov. Uniqueness and disclosure risk: Urn models and simulation. *Res. Official Statist.*, 4:23–40, 2001.

[19] S. E. Fienberg, U. E. Makov, and R. J. Steele. Disclosure limitation using perturbation and related methods for categorical data. *J. Official Statist.*, 14:485–511, 1998. With discussion.

[20] J. M. Gouweleeuw, P. Kooiman, L. C. R. J. Willenborg, and P. P. de. Wolf. Post randomization for statistical disclosure control: Theory and implementation. *J. Official Statist.*, 14:463–478, 1998.

[21] A. F. Karr, J. Lee, A. P. Sanil, J. Hernandez, S. Karimi, and K. Litwin. Disseminating information but protecting confidentiality. *IEEE Computer*, 34(2):36–37, 2001.

[22] A. F. Karr, J. Lee, A. P. Sanil, J. Hernandez, S. Karimi, and K. Litwin. Web-based systems that disseminate information but protect confidentiality. In A. K. Elmagarmid and W. M. McIver, editors, *Advances in Digital Government*. Kluwer, Amserdam, 2001. To appear.

[23] S. Keller–McNulty and G. T. Duncan. A progress report to the National Center for Education Statistics: Disclosure-limited statistical analysis of confidential data to support NSF-sponsored Digital Government grant. Technical Report LA-UR-01-1673, Los Alamos National Laboratory, 2001.

[24] S. L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, UK, 1996.

[25] J. Lee, C. Holloman, A. F. Karr, and A. P Sanil. Analysis of aggregated data in survey sampling with application to fertilizer/pesticide usage surveys. *Res. Official Statist.*, 4:101–116, 2001.

[26] H. G. Leimer. Optimal decomposition by clique separators. *Discrete Mathematics*, 113:99–123, 1993.

[27] D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *J. Amer. Statist. Assoc.*, 89:1535–1546, 1994.

[28] D. Madigan and J. York. Bayesian graphical models for discrete data. *Int. Statist. Rev.*, 63:215–232, 1995.

[29] National Institute of Statistical Sciences. Digital Government Project Web Site. Available on-line at www.niss.org/dg.

[30] S. Roehrig. Auditing disclosure in multi-way tables with cell suppression: Simplex and shuttle solutions. In *Proc. Joint Statistical Meetings*, Alexandria, VA, 1999. American Statistical Association.

[31] A. J. Saalfeld. Testing protection of suppressed cells in complex additive tables. Unpublished manuscript.

[32] S. M. Samuels. A Bayesian, species-sampling-inspired approach to the uniqueness problem in microdata disclosure risk assessment. *J. Official Statist.*, 14:373–383, 1998.

[33] Sun Microsystems, Inc. Java 2 Platform, Enterprise Edition Specification. Information available on-line at java.sun.com/j2ee/.

[34] Sun Microsystems, Inc. Java Servlet Technology. Information available on-line at java.sun.com/products/servlet/.

[35] R. E. Tarjan. Decomposition by clique separators. *Discrete Mathematics*, 55:221–232, 1985.

[36] The Apache Software Foundation. Jakarta Tomcat. Information available on-line at jakarta.apache.org/tomcat/.

[37] M. Trottini. A decision-theoretic approach to data disclosure problems. In *2nd Joint ECE/Eurostat Work Session on Statistical Data Confidentiality*, Luxembourg, 2001. Eurostat.

[38] M. Trottini. A decision-theoretic approach to data disclosure problems. *Res. Official Statist.*, 4:7–22, 2001.

[39] M. Trottini and Fienberg S. E. Modelling user uncertainty for disclosure risk and data utility, 2002. Submitted for publication.

[40] J. Whitakker. Fitting all possible decomposable and graphical models to multiway contingency tables. In T. Havraneek *et al.*, editor, *Compstat*, pages 401–406, Vienna, 1984. Physica-Verlag.

[41] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, New York, 1990.

[42] L. C. R. J. Willenborg and T. de Waal. *Statistical Disclosure Control in Practice*. Springer–Verlag, New York, 1996.

[43] L. C. R. J. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*. Springer–Verlag, New York, 2001.

[44] R. J. Yarger, G. Reese, and T. King. *MySQL and mSQL*. O'Reilly, Sebastopol, CA, 1999.