Exact and Heuristic Methods for Cell Suppression in Multi-Dimensional Linked Tables

George Duncan, Ramayya Krishnan, Rema Padman
Phyllis Reuther, Stephen Roehrig
H. John Heinz III School of Public Policy and Management
Carnegie Mellon University, Pittsburgh, PA 15213
Revision 1 for Operations Research

The increasing demand for information, coupled with the increasing capability of computer systems, has compelled information providers to reassess their procedures for preventing disclosure of confidential information. This paper considers the problem of protecting an unpublished, sensitive table by suppressing cells in related, published tables. A conventional integer programming technique for two-dimensional tables is extended to find an optimal suppression set for the public tables. This can be used to protect the confidentiality of sensitive data in three- and higher-dimensional tables. More importantly, heuristics are also presented to mitigate the computational difficulty of the integer program. An example is drawn from Federal Reserve Bank records. Data tables are randomly generated to assess the computational time/space restrictions of the IP model, and to evaluate the heuristics.

Under assurances of confidentiality, government agencies, commercial database organizations, and data warehouse managers collect and store multi-dimensional data about businesses and people. These same information organizations are under increasing demands to publish data compilations, e.g. tables of aggregate data, or to make specific data views accessible to interested subgroups of individuals within the collecting organization. In order to maintain confidentiality of respondents, data providers may suppress cell values if release would allow unauthorized inference of sensitive information (Cox 1980, Carvalho, Dellaert and Osorio 1994).

Inferential disclosure occurs when two or more data tables, taken together, enable a user to identify information pertaining to individual respondents even though no single table, taken by itself, is a direct disclosure (Fellegi 1972). This can occur, for example, when a linear combination of released cells results in unique identification of sensitive cell values. More subtly, there are situations in which the structure of the tables do not, *a priori*, assure disclosure, but some instances of the tables do. In Dutta Chowdhury et al. (1999), methods were presented by which certain non-sensitive projections of a database containing sensitive information can be combined to infer sensitive data. For example, two of the two-dimensional projections of an underlying three-dimensional table may be used to obtain bounds on the cells of the third, confidential, two-dimensional projection.

It is reasonable to ask to what extent disclosure is a problem and what characteristics are most apt to invoke it. In Dutta Chowdhury (1999), a formal probabilistic analysis demonstrated that, under reasonable assumptions about the distribution of the cell entries in the underlying 3-dimensional table, disclosure can indeed occur with non-trivial likelihood. While the incidence of disclosure decreases in proportion to table size, it increases in proportion to an increase in the probability of "unusual" values.

Once aware of a potential disclosure, it is important to prevent it (Duncan and Lambert 1989, Cox 1992). One method of protecting the confidential table is to suppress cells in the public tables (Causey, Cox and Ernst 1985, Lougee-Heimer 1989, Sullivan and Zayatz 1991, Fischetti and Salazar 2000, Kelly 1990, Guerts 1992, Kelly, Golden and Assad 1992, Zayatz 1992a, Zayatz 1992b, Cox 1995, Gopal, Goes and Garfinkel 1998). From the viewpoint of the publishers, suppressing all cells in all data views is obviously the most completely secure method of guaranteeing confidentiality, but it comes at the cost of not providing access to any of the data. A more practical solution would suppress only the minimum combination of cells in the public tables that assures adequate confidentiality. Other methods of preventing disclosure include controlled rounding (Cox 1987, Kelly, Golden and Assad 1990), random rounding (Nargundkar and Saveland 1993), data perturbation (Muralidhar, Batra and Kirs 1995), and the addition of random noise (Duncan and Mukherjee 1994, Evans Zayatz and Slanta 1996, Duncan and Mukherjee 2000).

In this paper, we adapt an existing integer programming model for finding an optimal cell suppression plan, but greatly improve its computational performance by using results from Dutta Chowdhury (1999). Then, much faster heuristics are presented which provide complete protection, but at the expense of potentially suppressing more cells than necessary. We give the results of computational experience with the IP and the heuristic techniques.

1 Inferential Disclosure

We illustrate the problem and our framework using a three-dimensional categorical database of financial data. In the general case of an n-dimensional database, some subset of projections of dimensions less than n are accessible, while the remainder are confidential.

All banks are required to file a quarterly Report of Condition with the Federal Reserve Bank (FRB). This report contains financial information including the number of outstanding loans of each type by days-past-due status. Regional FRB Data Services receives the report from each local bank, compiles the information, and releases aggregate statistical tables. Some of the micro-level information is confidential and disclosure would compromise the reporting bank (Board of Governors 1996).

Using this as our example, the three dimensions of the data are *Bank*, *Loan Type* (Real Estate (RE), Installment (IN), Credit Cards (CC), and Commercial (CM)), and *Debt Status* (Late 0–29 Days, Late 30–89 Days, Late 90+ Days, Non-Accrual). The Bank-Loan Type and Loan Type-Debt Status tables are not sensitive. They are publicly available from the Federal Reserve Board and are often used as indicators of a community's economic health. The Bank-Debt Status table is sensitive for loans past due 30 to 89 days. Many banks also prefer it not be publicly known to what extent they are carrying *any* past due loans. Therefore, the entire Bank vs Debt Status view is considered confidential.

A disclosure is detected if it is possible to infer either non-zero or exact values for any cell in the Bank vs Debt Status table. Hypothetical data are given in Figure 1. Note that in this figure, and throughout

Figure 1: Banking example information structure

the rest of the paper, we make the assumption that marginal values (row sums and column sums) are not provided as part of the data.

1.1 Linear Programming Formulation

The first problem to consider is whether inferential disclosure occurs in the confidential projection of the n-dimensional database. Dutta Chowdhury et al. (1999) developed linear programming (LP) methods for inferential disclosure detection in the sensitive data view from the contents of the publicly available, adjacent views of the underlying tables. Although we may not be able to directly determine the confidential cell entries, we can use this linear programming formulation to arrive at sharp bounds on the possible confidential values. These bounds may suggest that disclosure would occur if the public views were released; for example, when the upper and lower bounds are equal, the exact sensitive value is determined. A more subtle disclosure occurs when the bounds provide sufficient information to embarrass the data provider or respondent, such as might happen when non-zero lower bounds can be inferred.

Based on a three-dimensional table $X = [x_{ijk}]$, $i = 1, \ldots, I$, $j = 1, \ldots J$, $k = 1, \ldots, K$, the linear programming model computes lower and upper bounds for cells in the two-dimensional confidential table $C = [c_{i,k}]$, where $c_{ik} = \sum_j x_{ijk}$, from public tables $A = [a_{i,j}]$, $a_{ij} = \sum_k x_{ijk}$ and $B = [b_{j,k}]$, $b_{jk} = \sum_i x_{ijk}$ as follows:

$$\begin{aligned} \min / \max \quad & c_{ik} \\ \text{s.t.} \quad & \sum_k x_{ijk} = a_{ij}, \\ & (i = 1 \ldots I, \quad j = 1 \ldots J) \end{aligned} \tag{1}$$

$$\sum_k x_{jk} = b_{jk}, \tag{2}$$
$$(j = 1 \ldots J, \quad k = 1 \ldots K).$$

In this formulation, x_{ijk} are variables representing the unknown values of individual cells of the underlying categorical database. The terms c_{ik} denote the cell values of the confidential projection. The terms a_{ij} and b_{jk} denote the entries in the two two-dimensional, public views of the database. The LP formulation determines the upper and lower bounds on each cell (for each value of i and k respectively) of the confidential table C, subject to knowledge of the contents of the public tables.

1.2 LP Applied to the Banking Example

Using the linear programming formulation to obtain bounds for the confidential Bank vs Debt Status table is straightforward. Table 1 shows the (lower, upper) bound values for each cell. These bounds were obtained by solving the corresponding LP, i.e., minimizing and maximizing cell entries in the confidential table subject to the constraints given by the values in the two public tables.

	Late 0-29 days	Late 30-89 days	Late 90+ days	Non-accrual
National	(0,18)	(0,8)	(0,6)	(0,13)
Anytown	(0,31)	(0,14)	(0,21)	(0,20)
Iron City	(3,29)	(0,15)	(7,29)	(16,36)
1st Cyber	(0, 9)	(0,9)	(0,7)	(0,8)

Table 1: Results of LP Analysis: Bank vs Debt Status

In the Bank vs Debt Status table, it is reasonable to assume that banks would not want it publicly known if they carry any loans unpaid at due date. Since three of the sixteen entries in this table have a non-zero lower bound, disclosure has occurred.

In many data dissemination contexts, it is also considered a disclosure if a calculated upper or lower bound is too close to the actual sensitive cell value. In the remainder of the paper, we assume that the data producer will specify *desired* upper and lower bounds $\overline{D}$ and $\underline{D}$ for the sensitive table. For example, if it is felt that no computed upper bound should be closer than 20% of the actual value, and no computed lower bound be greater than zero, the desired bounds for the banking example are those shown in Figure 2. Comparing Figures 1 and 2 shows that the banking example contains an additional upper bound disclosure (Iron City Bank, non-accrual), for a total of four disclosures.

1.3 The MCA Algorithm

In addition to presenting the LP approach, Dutta Chowdhury et al. (1999) developed a matrix operator method called the MCA algorithm for finding sharp upper and lower bounds on confidential table entries. It is considerably faster than linear programming and provides insight into the construction of the heuristics we develop later for confidentiality protection. Additionally, it gives us a means for estimating some bounds

	Late 0-29 days	Late 30-89 days	Late 90+ days	Non-accrual
National	(0,15)	(0,1)	(0,2)	(0,2)
Anytown	(0,4)	(0,10)	(0,12)	(0,9)
Iron City	(0,25)	(0,4)	(0,18)	(0,37)
1st Cyber	(0, 2)	(0,2)	(0,4)	(0,1)

Table 2: Desired Bounds for Bank vs Debt Status

that are crucial to the success of integer programs for cell suppression. The algorithm relies on two matrix operators, defined next.

Definition 1 *Let $A = [a_{ij}]$ be an $I \times J$ matrix and $B = [b_{jk}]$ be a $J \times K$ matrix. The Cell-Maxima operator $\overline{\otimes}$ is a binary operator on (A, B) that yields an $I \times K$ matrix C^U defined by ik entries*

$$c_{ik}^U = \sum_j \min(a_{ij}, b_{jk}),$$

for $i = 1, \ldots, I,\ k = 1, \ldots, K$.

Definition 2 *With the same conditions as in Definition 1, the Cell-Minima operator $\underline{\otimes}$ is a binary operator on (A, B) that yields a matrix C^L of dimension $I \times K$ defined by ik entries*

$$c_{ik}^L = \sum_j \max(a_{ij} - \sum_{p \neq k} b_{jp}, 0),$$

for $i = 1, \ldots, I,\ k = 1, \ldots, K$.

These operators calculate bounds proven to be equivalent to those obtained by linear programming.

2 View Protection

If disclosure is detected, it becomes necessary for the database administrator to take protective action. While there are several methods available to facilitate table protection (e.g. rounding, noise addition, global recoding, and cell suppression), suppression has been argued to offer the most protection with the fewest theoretical problems (Willenberg, de Waal and Keller 1996). We note, however, that many of the other techniques are still being pursued (Evans, Zayatz and Slanta 1996; Duncan and Fienberg 1999).

The first, and most obvious, method to determine which cells could be suppressed is trial and error. For each cell with a disclosure, suppress some combination of the cells in the public views, then reapply the LP (or MCA) to see if the newly inferred table is now free of disclosure. It is desirable to minimize the number of cell suppressions. This creates a large search problem where, in the worst case, all possible combinations of cell suppressions, numbering $2^{(I*J)+(J*K)}$, will need to be explored. Even for small tables, say $3 \times 3 \times 3$, it means solving possibly as many as $2^{3*3+3*3} = 2^{18} = 262,144$ LP models. Later in this paper we will use a time-constrained version of this random search as a benchmark against which we compare a number of more disciplined procedures.

The extent of the disclosure problem and the exponentially increasing search space argue for a systematic approach to protecting the confidential data view. One solution discussed in this paper uses integer programming (IP) to determine the optimal set of suppressions for the public tables. For small tables, this provides an effective method that database administrators can use to eliminate inferential disclosure. For those situations where either large tables, or large numbers of tables, need to be audited and protected, we develop heuristic procedures which are fast and which guarantee protection, but which may not result in a guaranteed minimum number of suppressions.

A number of authors have considered the cell suppression problem for two-dimensional tables (Kelly, Golden and Assad 1992, Cox 1995 and their references), and several (e.g., Kelly 1990, Fischetti and Salazar González 2000) have extended the analysis higher-dimensional tables. The problem considered here, suppression in 2-dimensional marginal tables to protect a third, can be viewed as a special case of the general 3-dimensional cell suppression problem given arbitrary margins, where each cell has a suppression penalty associated with it. In our problem, the third (sensitive) 2-dimensional margin carries a penalty of zero, resulting in its full suppression[1]

3 Integer Programming Formulation

As previously mentioned, a database administrator can specify two additional tables of cell values, the upper and lower bound tables $\overline{D}$ and $\underline{D}$, representing the worst-case inferable values for the sensitive table C. If the value of a confidential cell can be inferred to be *within* these bounds, disclosure is considered to have occurred. These bounds might be set, for example, to $[0, \lceil c_{ik} + p\% \rceil]$, where p equals a percentage predetermined by the administrator. The Census Bureau uses a similar scheme for determining if a cell value violates their confidentiality policy; for them, the value of p is also confidential (Zayatz 1992). Including these tables, an integer programming model that finds an optimal set of cell suppressions to protect the confidential table can be written, along the lines of Kelly, Golden and Assad (1992).

We first present the complete IP, and then discuss it in detail. In what follows, $S^A = [s^A_{ij}]$ and $S^B = [s^B_{jk}]$ are arrays of indicator variables (1 indicating suppression), and L and U are 5-dimensional tables holding lower and upper bound values for X, respectively. We use "+" notation to denote marginal totals, e.g., $l_{+jkgh} = \sum_i l_{ijkgh}$.

$$\min \qquad z = \sum_{i=1}^{I}\sum_{j=1}^{J} s^A_{ij} + \sum_{j=1}^{J}\sum_{k=1}^{K} s^B_{jk} \tag{3}$$

s.t.

$$l_{ij+gh} \leq a_{ij} + M * s^A_{ij} \tag{4}$$

$$l_{ij+gh} \geq a_{ij} - M * s^A_{ij} \tag{5}$$

$$l_{+jkgh} \leq b_{jk} + M * s^B_{jk} \tag{6}$$

$$l_{+jkgh} \geq b_{jk} - M * s^B_{jk} \tag{7}$$

[1] We thank an anonymous referee for pointing out this interpretation.

$$
\begin{aligned}
u_{ij+gh} &\leq a_{ij} + M * s^A_{ij} && (8)\\
u_{ij+gh} &\geq a_{ij} - M * s^A_{ij} && (9)\\
u_{+jkgh} &\leq b_{jk} + M * s^B_{jk} && (10)\\
u_{+jkgh} &\geq b_{jk} - M * s^B_{jk} && (11)\\
l_{i+kgh} &\leq \underline{d}_{ik} && (12)\\
u_{i+kgh} &\geq \overline{d}_{ik} && (13)
\end{aligned}
$$

$$i, g = 1, \ldots, I, \quad j, h = 1, \ldots, J, \quad k = 1, \ldots, K$$

As mentioned earlier, a disclosure has occurred if at least one lower bound is non-zero, if a cell lower bound equals its upper bound, or more generally if the bounds generated by the LP are too tight to maintain confidentiality. The suppression set $\{S^A, S^B\}$ is feasible if the inferred lower and upper confidential cell bounds satisfy the desired tolerance levels. A feasible suppression set is optimal if it has a minimum number of nonzero elements.

The results from the IP formulation are the two tables S^A and S^B, which contain suggested cell suppressions for tables A and B. This solution allows inference from the published tables without disclosure of sensitive data. We now discuss the model in detail.

3.1 Variable Separation for Lower and Upper Bounds

We wish to determine that no configuration of table entries x_{ijk} will cause any of the desired lower or upper bounds to be violated. (The purpose of the additional subscripts g and h will be discussed shortly.) Additionally, the suppressed cells should be such that both lower and upper bounds should all be satisfied simultaneously. If we subject x_{ijk} to *both* the upper- and lower-bound constraints simultaneously, the IP will be infeasible; no configuration of the x_{ijk} can possibly give a confidential cell value both greater than an upper bound *and* less than a lower bound. To accomplish our task, then, for each cell in the three-dimensional table, we split the continuous variable x_{ijk} in the LP into two separate variables representing the lower and upper inferred cell values, l_{ijk} and u_{ijk}, respectively.

In addition to providing the cell values for the published tables, the database administrator can set desired lower and upper bounds for each cell in the confidential table using tables $\underline{D} = [\underline{d}_{ik}]$ and $\overline{D} = [\overline{d}_{ik}]$. Using the new variables, the desired bounds on the cell values in the sensitive table can then be formulated as

$$
\begin{aligned}
l_{i+k} &\leq \underline{d}_{ik} && (14)\\
u_{i+k} &\geq \overline{d}_{ik} && (15)
\end{aligned}
$$

$$i = 1, \ldots, I; \quad k = 1, \ldots, K.$$

3.2 Binary variables

If a cell is suppressed, the corresponding constraint in Equation (1) or (2) must be removed. In the IP model, removal of this suppression is accomplished by associating with each cell in the public tables the binary variables s^A_{ij} and s^B_{jk} which have the following definition: $s^A_{ij} = 1$ if cell a_{ij} is suppressed, 0 if released, and $s^B_{jk} = 1$ if b_{jk} is suppressed, 0 if released.

These binary variables are combined with the constraints from the LP model to enforce constraint suppression, as described in the next subsection. In the IP, only the binary variables are required to be integer valued, so the number of integer variables is equal to the number of cells in the public tables. The IP objective function (Equation 3) is the sum of these binary variables, that is, we seek to minimize the total number of suppressed cells.

3.3 Scalar Bounding

Each equation from the original LP model is converted to two inequalities in the IP model and combined with the binary variables defined earlier as follows:

$$l_{ij+} \leq a_{ij} + M * s^A_{ij} \tag{16}$$

$$l_{ij+} \geq a_{ij} - M * s^A_{ij} \tag{17}$$

$$i = 1, \ldots, I; \quad j = 1, \ldots, J,$$

where M is a sufficiently large positive weight. Both constraints are automatically satisfied when the cell is suppressed (i.e. $s^A_{ij} = 1$ or $s^B_{ij} = 1$). The remaining two lower constraints, L (summing over i for table B) and the four U constraints follow the same pattern. When the binary suppression variables are zero, the two inequalities revert to the equation for the corresponding published cell value.

So-called "big M" methods such as this one are often criticized (Bradley, Hax and Magnanti 1977) for their proclivity for causing numerical instability. Our empirical evidence is that, for the problems we have solved, instability is not a problem. An additional criticism is that choosing M to be a very large value can make solving the model using a branch and bound algorithm extremely time consuming. We want to choose M as small as possible so that the bounds produced in the LP relaxation stage of branch and bound are as sharp as possible. We use insights from the MCA algorithm to find suitable values for the M parameters. We defer the analysis generating the M values until §5, since it relies heavily on the heuristics we present there. However, we remark here that proper choices can reduce the computation time of the IP by roughly two orders of magnitude, allowing us to routinely solve problems with hundreds of suppression variables in a few minutes on a PC.

3.4 Completing the IP Formulation

It remains to describe the function of the additional subscripts g, h in the IP formulation given at the beginning of this section. These new indices have the same ranges as the indices i and k respectively; their purpose is to provide separate "copies" of the original l_{ijk} and u_{ijk} variables for each cell in the confidential table. The new indices vary in step with i and k, so the new continuous variables are l_{ijkgh} and u_{ijkgh} (for $i = 1, \ldots, I, j = 1, \ldots, J, k = 1, \ldots, K, g = 1, \ldots, I, h = 1, \ldots, K$), respectively, for the lower and upper confidential cell tables.

Without these additional indices, the constraints would still restrict the entries in the underlying 3-D table to be consistent with the released data after supression; however, the inferred cells must respect all the desired upper and lower bounds simultaneously. This "table-wise" formulation is appropriate if concern is with the class of functions having maxima and minima that are functions of only the set of maxima and minima of their arguments. Linear combinations, for example, are in this class. However, data administrators are primarily interested in ensuring that each individual cell, rather than functions of cell values, are protected. The addition of the two subscripts g and h, while increasing the total number of variables, brings the model into conformance with this latter requirement.

3.5 Example Results

The IP model for the Banking example was solved using $\underline{d}_{ik} = 0$, $\overline{d}_{ik} = \lceil c_{ik} + 20\% \rceil$. The suppression pattern was the following.

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & - \\ 3 & 16 & 17 & 35 \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & 9 \\ 12 & 4 & 12 & 2 \\ - & 5 & 0 & - \\ 10 & - & 14 & 15 \end{bmatrix}$$

The new lower and upper bounds were inferred from the published tables using the LP formulation while suppressing these four cells. The new sensitive view is:

	Late 0-29 days	Late 30-89 days	Late 90+ days	Non-accrual
National	(0,18)	(0,8)	(0,6)	(0,13)
Anytown	(0,35)	(0,∞)	(0,29)	(0,30)
Iron City	(0,42)	(0,47)	(0,29)	(0,37)
1st Cyber	(0, 9)	(0,9)	(0,7)	(0,8)

Table 3: IP Suppressions: Bank vs Debt Status

As an example of the value of minimizing M in equations 4–11, the IP and its relaxation were solved for several values of M. We initially set $M = 2000$ for each constraint. The nonzero terms in the relaxed solution were on the order of 0.005. Solving the IP for this choice of M required approximately 25,000 iterations. When the value of M was computed individually, as in §5.2, the nonzero relaxation values were typically 0.2 or greater. The effect of this on the IP solution process was substantial, reducing the number of iterations required to 2841, close to an order of magnitude improvement. Larger tables typically showed an even greater improvement.

Integer programming techniques for cell suppression typically consume computer processing time exponentially with respect to table size, which makes them impractical for large tables. For two-dimensional problems reported in the literature, "large" has been considered something on the order of $N \geq 13$ (Kelly 1990, Sullivan and Zayatz 1992), although Fischetti and Salazar González (2000) have extended this considerably . For the three-dimensional problems considered here, $N \geq 13$ is still a reasonable estimate, but note that N refers to the size of each of three dimensions, rather than two. Larger problems consume large amounts of memory (for both the continuous variables and the data structures for organizing the integer programming logic), effectively becoming impossible to run on most commercially available computers and software without extending the default timing parameters, effectively trading time for space. Later in this paper we give examples of the computer resources required for the IP; with current technology, $5 \times 5 \times 5$ problems typically take many days to solve without the big-M improvements presented here. With these improvements, a $10 \times 10 \times 10$ problem can be solved in less than an hour on a PC (and often in minutes).

4 Discussion

In Dutta Chowdhury et al. (1999), a pair of matrix operators (MCA) was presented which allows a database administrator to detect disclosures which might result from the publication of multiple linked tables. With this method, an administrator can *evaluate* a data access policy; that is, it becomes easy to check whether institutional rules regarding release of aggregate tables actually do protect the data sources. However, while the LP gives sharp bounds on the values of confidential cells, it provides no guidance on what to do if disclosure is detected. Alternate policies could be compared, but at best this would be an *ad hoc* approach.

In this paper it has so far been shown that a conventional IP formulation, augmented with insights suggested by the MCA algorithm, provides a systematic way in which database administrators can not only detect disclosure, but also find an optimal set of cell suppressions to protect pairs of linked data tables. This approach permits the administrator to specify, at the level of the individual confidential cells, appropriate protection. The IP then operates autonomously to find the minimal number of suppressions which provide this protection.

While it hasn't been discussed here, the method is easily generalized to base tables of higher dimension, and to views numbering greater than two. The work by Dobra and Fienberg (199) on graphical models specifies conditions under which the disclosure detection problem for released marginals decomposes into simpler pieces. With increasing availability, especially over the Internet, of such higher-dimensional data tables, disclosure prevention methods designed only for two-dimensional tables must be re-thought and extended.

It can be easily seen that many variations on the method presented here are possible. For example, the optimality criterion which says "fewer suppressions are better" might, in some circumstances or subject areas, be changed. One obvious alternative would be to minimize the weighted sum of the suppressed cell entries, or more generally, to minimize some utility function defined over the cells of the published tables. Another policy might be to "balance" the suppressions over the published tables, so that each table be maximally informative in its own right.

It is clear, however, that the IP approach—even if augmented to include refined values of M— presented here can indeed become very difficult or impossible to carry out with large tables. This is balanced to an extent by the experimental evidence that larger tables generally give rise to fewer disclosures. However, faster methods which guarantee protection while possibly over-protecting are valuable in those situations where many tables, or large tables, need to be made safe. We now turn to the development of heuristics based on the MCA algorithm given in §1.3.

5 Heuristics for Cell Suppression

The integer programming formulation presented above provides an exact solution to the suppression problem. However, it is obvious that the time to find optimal solutions is unacceptable if one has large tables. In this section, we examine two heuristic procedures that, while not guaranteed to find optimal suppression patterns, do intelligently examine the space of such patterns.

The heuristics considered here attempt to take advantage of the structure of the suppression problem. One way of defining that structure is through the IP formulation described earlier. The analysis approach leading to the MCA algorithm yields another perspective on the structure of the problem. We will present and evaluate two approximate algorithms, derived from MCA, that again guarantee feasible suppressions. The first of these is a straightforward greedy approach, and the second is a tabu search whose search "directions" amount to addition or deletion of suppressions.

Our last heuristic is a baseline check on the rest. It randomly assigns suppression cells, and checks via the MCA algorithm to see if the current suppression pattern is feasible. The best suppression pattern obtained over some fixed number of iterations of random assignment is logged. At each iteration, the cost of "determining" the suppressions is small, and the cost of checking the protection outcome is also small, given the speed of the MCA algorithm.

5.1 A Greedy Heuristic Based on MCA

In this section we look more carefully at the MCA algorithm, note how it may be modified to account for cell suppressions, and then use these observations to suggest a simple procedure for actually determining cell suppressions. The result is an algorithm that guarantees protection of sensitive cells but which may not be optimal in terms of minimizing the number of cells suppressed. In addition, the analysis suggests good values for the M parameters used earlier in the IP formulation.

Consider once again the banking example from §1. The public views are reproduced for convenience in Figure 4.

	RE	I	CC	CM
National	14	1	3	0
Anytown	12	10	3	6
Iron City	3	16	17	35
1st Cyber	3	3	2	1

	Late 0-29 days	Late 30-89 days	Late 90+ days	Non-accrual
RE	14	4	5	9
I	12	4	12	2
CC	4	5	0	16
CM	10	3	14	15

Table 4: Banking Example: Public Views

The upper bound operator $\overline{\otimes}$ works much like ordinary matrix multiplication, except the *minima* of cell value pairs from the two tables are summed, rather than their *products*. The upper bound confidential cell entry for National-Late 0-29 Days is thus computed as

$$\begin{aligned} c_{11}^U &= \min(14,14) \\ &+ \min(1,12) \\ &+ \min(3,4) \\ &+ \min(0,10) = 18. \end{aligned}$$

The lower bound operator $\underline{\otimes}$ is somewhat different. Its addends are the larger of zero and the difference between a cell in the first table and the sum of complementary cells in the second. For National-Late 0-29 Days, the sum is

$$\begin{aligned} c^L_{11} &= \max(0, 14 - (4 + 5 + 9)) \\ &+ \max(0, 1 - (4 + 12 + 2)) \\ &+ \max(0, 3 - (5 + 0 + 16)) \\ &+ \max(0, 0 - (3 + 14 + 15)) = 0 \end{aligned}$$

Now consider the situation where a cell is suppressed. In the case of $\overline{\otimes}$, we sum terms of the form $\min(a_{ij}, b_{jk})$. If either (but not both) of a_{ij} or b_{jk} is suppressed, its value is unknown, but could potentially be very large. Thus the min should be taken to be the remaining (unsuppressed) value, because we are looking for the largest value this term could take on. If both a_{ij} and b_{jk} are suppressed, the min is unbounded, so this term, and thus the entire sum, is unbounded. It can be seen, then, that suppressing a cell has the potential to increase the upper bound on any confidential cell which relies on its value. The amount of increase can be at most the difference $|a_{ij} - b_{jk}|$. Of course, if the suppressed cell was in fact the larger of the two compared by the min operator, the upper bound is unchanged.

For the lower bounding operator $\underline{\otimes}$, a similar result holds. In this case, the effect of a suppression is to drive the lower bound downward. Any term in the sum is of the form $\max(a_{ij} - \sum_{p \neq k} b_{jp}, 0)$. If a_{ij} is suppressed, its actual value may be as small as zero. Thus $a_{ij} - \sum_{p \neq k} b_{jp}$ is nonpositive, so the term must be taken as zero. Conversely, if some $b_{jp}, k \neq p$ is suppressed, it could well have been very large, and thus the term must again be taken as zero. The effect of either of these suppressions is to reduce the overall lower bound by any positive contribution made by the term.

The result of this analysis is that lower and upper bounds may be altered in discrete steps by judicious choices of suppressed cells. We next describe a simple greedy heuristic to select cell suppressions. As before, let $A[ij]$ and $B[jk]$ be the non-confidential tables, and let $\overline{D}[ik]$ and $\underline{D}[ik]$ be the desired upper and lower bounds on the inferred confidential table $C[ik]$.

One way of dealing with an upper bound disclosure at (i, k) would be to successively suppress cells in, say, row i of A, until the MCA-calculated value of c_{ik} exceeded that of $\overline{d}_{ik}$. That task accomplished, one would need to ensure that no suppressed value in A, in column j, say, could be calculated from the jth row of B. Recall that the column sums of A must equal the corresponding row sums of B. Since in this paper we assume that the row and column marginal totals are not published, such a calculation can only be done when all cells in row j of B are unsuppressed. If this is the case, an additional suppression in row j of B is required. One might choose a cell arbitrarily.

A more efficient method, and the one used in the greedy heuristic, is to suppress the pair (a_{ij}, b_{jk}) for some j. These two suppressions, by the logic presented above, are sufficient to drive the upper bound of c_{ik} to infinity, and at the same time eliminate the possibility of calculating a suppressed cell in one table from a corresponding row or column in the other. In practice, one might choose j to minimize the sum of the suppressed cells, if this proved to be an important secondary consideration.

For a lower bound disclosure at c_{ik}, a very simple procedure guaranteed to drive the lower bound below $\underline{d}_{ik}$ can be used. First, the (positive) difference between the MCA-calculated value of c_{ik} and $\underline{d}_{ik}$ is found. Then for a row j in B that contributes a positive lower bound component p, a_{ij} is suppressed. The difference is decremented by p, and this process is repeated until the difference is zero or less. Then complementary suppressions in B are chosen as necessary.

Pseudocode for this algorithm is given in Figure 2. The result of this elementary procedure, applied to the banking example, is the following, where again a cell entry "–" indicates a suppression.

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & 6 \\ - & 16 & - & - \\ 3 & 3 & 2 & 1 \end{bmatrix} \qquad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & - \\ 12 & 4 & 12 & 2 \\ 4 & 5 & - & 16 \\ 10 & - & 14 & 15 \end{bmatrix}$$

5.2 Choosing Values for "Big M"

The foregoing discussion of the MCA algorithm also suggests a way to optimally choose values for the M parameters used in the integer programming formulation. Specifically, it shows exactly how much of an increase in u_{ij+} or u_{+jk} (or decrease in l_{ij+} or l_{+jk}) can result from the suppression of any corresponding a_{ij} or b_{jk} cell.

Figure 3 presents one component of the banking example in network form. Because of the decomposable nature of the problem, there will be four such components, one for each value of index j. The component for $j = 4$ is shown, and the values of the nodes are taken from the original, unsuppressed tables. Each node on the left of the network corresponds to an entry in A_{ij}, while those on the right correspond to entries in B_{jk}. Arcs can be thought of as carrying "flows" of loans from banks to debt status.

Consider first the contributions to lower bound violations apparent in this graph. Following the MCA lower bound procedure, node i_3j_4, if suppressed, will cause a lower bound decrease of 3 for confidential cell i_3k_1, and similar decreases of 7 and 8 for i_3k_3 and i_3k_4 respectively. Thus its suppression can cause a maximum decrease of 8. It's easy to check that the suppression of any other cell (or combination of cells) in this graph cannot reduce a lower bound by more than 8. Consequently, node i_3j_4 need source no more than 8 more units (over and above its current value of 35) as a result of its suppression. Thus we refine the M parameters, tailoring them to the subgraphs they are associated with. That is, we introduce the refinements M_j^{LB}, and these are used in the suppression constraints associated with l_{ij+} and l_{+jk}. We have just determined that $M_4^{LB} = 8$, and a similar analysis finds that $M_1^{LB} = 0$, $M_2^{LB} = 0$ and $M_3^{LB} = 8$.

These new values for the M allow the variables l_{ijk} sufficient freedom to drive the confidential cells l_{i+k} down to achieve the desired lower bounds. Since we use separate copies of the l_{ijk} for each potential suppression cell (i.e., l_{ijklm}), we needn't worry about interactions between disclosures. Note also that in those cases where $M_j^{LB} = 0$, the $\geq/\leq$ constraint pairs revert to the original equality constraints. This effectively removes $I \times K$ binary variables from the formulation, increasing performance considerably.

Identifying M values for the upper bound constraints is somewhat similar, but necessarily less precise. Looking again at Figure 3, and following the MCA algorithm for upper bounds, the potential effects of suppressing any one node are readily apparent. However, when nodes are suppressed in combination, the result may be to move an upper bound to infinity. Nonetheless, we know, from the differences between the MCA upper bounds and the desired upper bounds, how much much change is required (the UGAPs in the greedy heuristic presented earlier). Thus we introduce new constants M_j^{UB}, each of which is equal to the maximum of the UGAPs. Once again, since there are separate copies of the u_{ijk} for each potential suppression cell, interactions needn't be considered.

Finally, it is clear that the upper bound for an ik cell can be driven to infinity in *any* of the subgraphs by simply choosing an ij, jk pair of suppressions in that subgraph. This implies that if there are lower bound

INPUT: $A[i,j], B[j,k], \underline{D}[i,k], \overline{D}[i,k]$
OUTPUT: $S^A[i,j], S^B[j,k]$

$\overline{C}[i,k] = A \mathbin{\overline{\otimes}} B, \quad \underline{C}[i,k] = A \mathbin{\underline{\otimes}} B$
$s^A_{ij} = 0, \quad s^B_{jk} = 0$

\\\\ Upper bounding
FOR each i, k s. th. $\overline{c}_{ik} < \overline{d}_{ik}$ {
    IF ($\exists j$ s. th. $s^A_{ij} = 1$ AND $s^B_{jk} = 1$)
        \\\\ do nothing
    ELSE IF ($\exists j$ s. th. $s^A_{ij} = 1$)
        $s^B_{jk} = 1$
    ELSE IF ($\exists j$ s. th. $s^B_{jk} = 1$)
        $s^A_{ij} = 1$
    ELSE
        $s^A_{i1} = 1, \quad s^B 1k = 1$
}

\\\\ Lower bounding
FOR each i, k s. th. $\underline{c}_{ik} > \underline{d}_{ik}$ {
    LGAP = $\underline{c}_{ik} - \underline{d}_{ik}$
    JIndex = 1
    WHILE LGAP > 0
        DIFF = $(1 - s^A_{i,\mathrm{JIndex}}) * A_{i,\mathrm{JIndex}} - \sum_{j \neq \mathrm{JIndex}} b_{jk}$
        IF (DIFF > 0)
            $s^A_{i,\mathrm{JIndex}} = 1$
            LGAP = LGAP - DIFF
}

\\\\ Finding complementary suppressions
FOR each j {
    IF $\sum_i s^A_{ij} + \sum_j s^B_{jk} < 2$
        IF $\sum_i s^A_{ij} = 0$
            $s^A_{1,j} = 1$
        ELSE
            $s^B_{j1} = 1$
}

Figure 2: Pseudocode for the Greedy Heuristic

disclosures, the best suppression choices for upper bound disclosures will be in those subgraphs that can have an effect on the lower bounds, since a required lower bound suppression can be "reused". Since minimization of the total number of suppressions is the overall goal, no suppressions to combat upper bound disclosures will be chosen in subgraphs that do not also decrease lower bounds. In the banking example, the subgraphs for $j = 1$ and $j = 2$ cannot contribute to lower bound decreases, so it's possible to set M_1^{UB} and $M^U B_2$ to zero. Again, this effectively forces the binary suppression variables associated with these subgraphs to zero.

The cumulative effect of choosing the M values according to this plan is to make their values as small as possible, and in some instances, remove large numbers binary variables from the IP formulation. In tests on randomly generated problems (presented in §6), there was a dramatic decrease in IP solution times, and as a consequence much larger problems could be solved to optimality.

5.3 A Tabu Search based on MCA

Tabu search (Glover and Laguna 1997) is a way of moving through the space of possible suppression patterns for tables A and B. It is similar to greedy searches, but it attempts to overcome the problem of getting stuck at a local, rather than global, optimum. To do this, some search directions are labeled "tabu," that is, off limits. For instance, if the search has found a local optimum, it may be wise to back off from it, and label the direction back to the local optimum tabu. This forces the search to move in other, possibly less immediately gratifying, directions. A tabu direction does not remain tabu forever; the thinking is that after the search has moved some distance from the spot where the direction was made tabu, it may again be used, since it will now not lead to the original dead-end.

For the suppression problem, the search moves through suppression patterns. So, a "direction" can be thought of as suppressing a new cell (and possibly unsuppressing another). A tabu search direction can then be indicated by disallowing the suppression of a particular cell.

To keep track of tabu cells, let $T^A[i, j]$ and $T^B[j, k]$ be tables containing the "tabu status" of each cell in A and B. These tables will be initialized to zero, but when an A or B cell becomes tabu, the corresponding entry in T^A or T^B will be set to a positive integer corresponding to the "tabu tenure" (10 in our implementation). Thereafter, values in T^A and T^B will be decremented (according to the policy that follows) until they again reach zero, at which time the A and B cells they refer to are no longer tabu.

The starting point for the tabu search is the result of applying one iteration of a random heuristic (§5.5) to the tables A and B. The tabu search then iterates over the following sequence of steps.

1. *Identify redundant suppressions.* For each pair (column j of A, row j of B), if the sum of suppressions is greater than two, pick a suppression at random, unsuppress it (giving back its original value), and check for disclosure via MCA. If the result is feasible (no disclosures), leave the cell unsuppressed, and mark it as tabu. In our example, there are no redundancies.

2. *Swapping cells.* Swap a suppressed cell with an unsuppressed cell in the same column (of A) or row (of B), provided it is feasible. In our example, swapping cells b_{33} and b_{34} (that is, unsuppressing b_{33} and suppressing b_{34}) leaves the tables feasible. Mark the cell that was swapped out (b_{33}) tabu by setting

t_{33}^B to the value of the tabu tenure. The result is

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & 6 \\ - & 16 & - & - \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & - \\ 12 & 4 & 12 & 2 \\ 4 & 5 & 0 & - \\ 10 & - & 14 & 15 \end{bmatrix}$$

A second application of this rule (in a following iteration) might swap a_{34} and a_{24}. This would leave the suppressed tables as

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & - \\ - & 16 & - & 35 \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & - \\ 12 & 4 & 12 & 2 \\ 4 & 5 & 0 & - \\ 10 & - & 14 & 15 \end{bmatrix}.$$

3. *Eliminate unnecessary column/row pairs.* It may happen that as a result of previous changes, a *pair* of suppressed cells is no longer necessary. Since the rule is that we must have the sum of suppressions in a column/row pair greater than two, or have none at all, we might look to see if "none at all" will also work.

   For any column of A with two or more suppressions, or any row of B with two or more suppressions, or any column/row pair with two or more suppressions, remove a pair of suppressions and check for feasibility (via MCA). If feasible, mark the eliminated suppressions as tabu.

   In the example, the pair a_{31} and b_{14} are redundant, so they may be eliminated. This leaves us with two fewer suppressions, in fact bringing us to the minimum provided by the integer programming formulation. Here is the result.

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & - \\ 3 & 16 & - & 35 \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & 9 \\ 12 & 4 & 12 & 2 \\ 4 & 5 & 0 & - \\ 10 & - & 14 & 15 \end{bmatrix}. \tag{18}$$

   Compare this with the integer programming solution:

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & 3 & - \\ 3 & 16 & 17 & 35 \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & 4 & 5 & 9 \\ 12 & 4 & 12 & 2 \\ - & 5 & 0 & - \\ 10 & - & 14 & 15 \end{bmatrix}.$$

It is immediately apparent from this comparison that an optimal suppression solution need not be unique. The tabu solution in this instance has the property that the suppressions are evenly allocated over A and B, although this is not guaranteed. As mentioned earlier, a modified IP objective function could include this "balance" as part of its optimality criterion.

5.4 Tabu Search Algorithm

The following steps implement the overall tabu search algorithm.

1. Run one iteration of the random algorithm, then initialize data structures.
2. Iterate over the following steps, for some reasonable number of iterations (500 in our tests).
   - Generate a candidate *improving* move (i.e., 1 or 3). If one exists, apply it and decrement the status of any tabu cells. Iterate.
   - If no improving move exists, identify a swap. If one exists, apply it and decrement the status of any tabu cells. Iterate.
   - If no such moves are available, identify an improving move that uses a tabu cell (the "aspiration criterion"). If one exists, apply it and decrement the status of any tabu cells. Iterate.
   - If a candidate is still not found, generate a new starting configuration by running a single iteration of the random procedure described below. Iterate.
3. Return the results.

When applied to the banking example, the results produced by this algorithm were those of Equation 18.

5.5 Random Suppression

The baseline strategy for determining suppressions is a simple random procedure. Since MCA is quite fast, it is plausible that a generate-and-test process might find reasonable suppression patterns in a time much shorter than the optimal IP.

In our approach, all cells in A and B are initially unsuppressed. A cell is drawn at random from A. It is then suppressed, and MCA applied to determine if this protection is sufficient. If so, the procedure ends. If not, another cell, this time from B is randomly chosen and suppressed, and MCA is applied again. This process continues, alternating from A to B until the protection criterion is met. Then a check is made to ensure that no suppression cell value in A (resp. B) can be calculated from a corresponding row in B (resp. column in A). If such a cell can be calculated, an additional suppression in B (resp. A) is chosen. The suppression pattern is logged, and then the whole procedure is begun anew. If, at any succeeding iteration, a pattern with fewer suppressions than the previous best is found, it becomes the new best pattern. For the banking example, the following result, having six suppressions, was obtained.

$$A[i,j] = \begin{bmatrix} 14 & 1 & 3 & 0 \\ 12 & 10 & - & 6 \\ - & 16 & 17 & - \\ 3 & 3 & 2 & 1 \end{bmatrix} \quad B[j,k] = \begin{bmatrix} 14 & - & 5 & 9 \\ 12 & 4 & 12 & 2 \\ 4 & - & 0 & 16 \\ 10 & - & 14 & 15 \end{bmatrix}$$

This result is inferior to both the IP and tabu solutions. Note, however, that one cannot know if the next random solution (if it were generated) might be better. With both the tabu and random algorithms, resources are allocated; when they are exhausted, the best solution is simply accepted.

6 Comparison of Heuristics and Exact Solutions

Two types of computational experiments were performed to compare the heuristics with the integer programming model. In the first, a small sample of randomly generated problems was submitted to the IP and all three heuristics. This sample was necessarily small because of the execution times of the IPs. In the second set of experiments, the heuristics were compared on a larger randomly generated sample.

The sample problems were generated as follows. Since the probabilistic analysis in Dutta Chowdhury (1999) showed that disclosures were more likely in base tables with extreme values, cell entries in $n \times n \times n$ tables were drawn from a distribution with values in the sets $\{0, 1, 2, 3\}$ and $\{500, 501, \ldots 1000\}$. With probability 0.9, a value was drawn uniformly from $\{0, 1, 2, 3\}$; otherwise the 3-dimensional cell value was drawn uniformly from $\{500, 501, \ldots 1000\}$. For each 3-D table so generated, the projections A, B, and C were computed. The MCA algorithm was then used to calculate upper and lower bounds on C, and these were compared with $\underline{D}$ and $\overline{D}$ to check for disclosures. Each $\underline{d}_{ik}$ was set to zero, while $\overline{d}_{ik}$ was set to $1.2c_{ik}$. If at least one upper bound disclosure and at least three nonzero lower bound disclosures occurred, the problem was retained for use in the computational experiments. Otherwise, the problem was discarded.

The IP and the greedy algorithms are deterministic, while both the tabu and random procedures are inherently stochastic, and require a stopping criterion, which may be given as a number of iterations or a fixed time allotment. In the comparisons that follow, execution times for the IP and greedy procedures are those required for them to run to completion. For the tabu search, 500 iterations were run, and the times and best results noted. To fairly compare the random heuristic with the tabu search, the numbers of iterations of the random search were set for each table size so that its execution times were at least as long, on average, as those of the tabu search.

Figure 5 gives a summary of IP solutions and the effort and space required to obtain them. We used an AMPL formulation that was a straightforward transcription of the cell-wise formulation. Rather than show actual times, the number of iterations required by the CPLEX solver are given. On a 300 MHz Pentium PC, running times for the IPs ranged from a few seconds to a maximum of 20 minutes for a $10 \times 10 \times 10$ problem. For the larger problems shown here, and surely for even larger problems encountered in actual use, one difficulty is that the IP formulation includes extremely large numbers of continuous variables ($>$ 200K for the $10 \times 10 \times 10$ problems). Thus the time taken in simplex iterations used in the IP branch and bound procedure, and the heuristics that CPLEX uses in trying to get an initial feasible integer solution, become an additional limiting factor. Indeed, with the method described in Section 5.2 for estimating the M parameters, CPLEX's heuristic was often able to find an optimal integer solution without having to resort to branching.

Figure 5 also shows the results of the three heuristics on the same problems. It can be seen that the greedy heuristic produces results that are on par with the random procedure. The tabu search was best, achieving optimal solutions in all but three of the 35 cases tested.

Table 6 compares actual execution times and average number of suppressions produced by the three heuristics, on a sample of 1000 problems of each size listed. All testing was done on a 300 MHz Pentium II, using code written in C++. The greedy heuristic is fastest, and for the smaller problems, produces solutions generally inferior to both the random and tabu procedures. However, as the problem size increases, the greedy and random procedures trade positions. The tabu search again produces the best results, clearly dominating for all problem sizes.

	Disclosures		IP			Greedy	Random	Tabu
Problem	UB	Nonzero	Iter.	Branchs	#S	#S	#S	#S
4x4-1	5	4	1K	0	8	12	11	8
4x4-2	4	3	2K	18	6	10	9	6
4x4-3	5	4	3K	20	6	11	10	6
4x4-4	5	4	4K	21	7	12	10	7
4x4-5	5	4	1K	0	8	10	10	8
5x5-1	6	7	8K	5	8	13	13	8
5x5-2	4	3	7K	32	6	11	8	6
5x5-3	6	3	74K	899	9	12	16	9
5x5-4	8	7	115K	670	8	14	15	8
5x5-5	6	4	21K	123	8	12	13	8
6x6-1	5	5	11K	82	7	12	16	7
6x6-2	5	4	38K	90	6	12	12	7
6x6-3	5	6	4	0	4	6	6	4
6x6-4	1	3	3K	3	8	12	13	8
6x6-5	6	4	21K	123	4	6	6	4
7x7-1	4	3	2K	0	6	7	15	6
7x7-2	1	4	4K	0	4	5	10	4
7x7-3	2	3	6K	0	6	7	12	6
7x7-4	4	3	28K	49	6	9	10	6
7x7-5	1	4	5K	0	4	7	8	4
8x8-1	2	3	4K	0	4	8	4	4
8x8-2	2	3	5K	0	3	6	8	3
8x8-3	1	3	8K	0	4	4	8	4
8x8-4	2	3	20K	17	4	8	9	4
8x8-5	1	3	5K	0	2	2	2	2
9x9-1	3	4	8K	0	4	7	16	5
9x9-2	6	4	5K	0	3	12	19	6
9x9-3	1	3	8K	0	2	2	4	2
9x9-4	3	3	8K	0	4	8	9	4
9x9-5	2	3	7K	0	4	4	8	4
10x10-1	2	3	32K	14	5	7	14	5
10x10-2	4	3	12K	0	5	7	20	5
10x10-3	2	3	10K	0	4	6	11	4
10x10-4	3	3	10K	0	5	5	10	5
10x10-5	2	3	10K	0	3	3	6	3

Table 5: Comparison of Heuristics With IP Solutions

	Avg. Disclosures		Greedy		Random		Tabu	
Table Size	UB	Nonzero	Time	#S	Time	#S	Time	#S
$4 \times 4 \times 4$	3.94	4.4	< 0.01	11.4	0.43	9.40	0.43	5.58
$5 \times 5 \times 5$	5.78	4.41	< 0.01	13.7	0.72	13.0	0.72	7.49
$6 \times 6 \times 6$	6.34	4.25	< 0.01	14.4	1.05	15.8	1.05	7.37
$7 \times 7 \times 7$	6.11	3.90	< 0.01	13.7	1.41	17.8	1.41	6.99
$8 \times 8 \times 8$	5.12	3.54	< 0.01	11.8	2.01	18.0	1.79	6.16
$9 \times 9 \times 9$	4.20	3.34	< 0.01	9.90	2.71	18.0	2.25	5.34
$10 \times 10 \times 10$	3.37	3.24	< 0.01	8.26	3.66	16.6	2.75	4.53

Table 6: Results of the Three Heuristics on Sets of 1000 Randomly Generated Problems

7 Summary

In this paper, we have considered the problem of cell suppression to protect a table of sensitive data. For published 2-dimensional tables, this is a well-researched, though still difficult, problem. What is new here is that the sensitive table itself is *not* published. Rather, related tables from which one might infer values in the sensitive table *are* published, and it is these tables that require suppressions to protect against such inference. An example from the Federal Reserve Bank reporting system illustrated one situation in which linking individually innocuous (but closely related) tables can pose a threat to the confidentiality of an unpublished sensitive table.

An IP model, discussed in Section 3.4, provides solutions that protect the sensitive table while minimizing the total number of suppressions in the published tables. A method for determining good choices for the big-M parameters was given, which greatly extends the size of problems that may be solved to guaranteed optimality. Nonetheless, larger problems can very time-consuming to solve. To meliorate this problem, two heuristics based on the matrix operators presented in Dutta Chowdhury (1999) were developed. The first, a simple greedy algorithm, is extremely fast, and gives complete protection, though it seldom finds a solution having the minimum possible number of suppressions. The second, based on the notion of tabu search, is still quite fast, and in our tests has performed as well as integer programming in most cases. Both heuristics proved to be markedly superior to a random procedure that produces secure suppression patterns but does not consider the linked structure of the published tables.

8 Acknowldgements

This research was supported in part by the National Science Foundation, NSF IRI-9312143, and by the U.S. Army Research Office under grant DAAH04-94-6-0239. We thank Anthony Colatrella for invaluable help in programming the heuristics and collecting the statistics presented here. We also thank three anonymous referees for their helpful comments, especially those that led to the development of the Big-M improvements discussed in § 5.2.

9 References

BOARD OF GOVERNERS OF THE FEDERAL RESERVE SYSTEM 1996. A Consolidated Report of Condition and Income for A Bank With Domestic Offices Only and Total Assets of Less Than $100 Million - FFIEC 034". Federal Deposit Insurance Corporation, Office of the Comptroller of the Currency, Washington, D.C..

CAUSEY, B.D., L.H. COX AND L.R. ERNST 1985. Application of Transportation Theory to Statistical Problems. JASA 80, 903–909.

COX, L.H. 1980. Suppression Methodology and Statistical Disclosure Control. JASA 75, 377–385.

COX, L.H. 1987, A Constructive Procedure for Unbiased Controlled Rounding. JASA 82, 520–524.

COX, L.H. 1992. Solving Confidentiality Protection Problems in Tabulations Using Network Optimization: A Network Model for Cell Suppression in U.S. Economic Censuses. International Seminar on Statistical Confidentiality, Eurostat, Dublin, Ireland, 229–245.

COX, L.H. 1995. Network Models for Complementary Cell Suppression. JASA 90, 1453–1462.

DE CARVALHO, F.D., N. DELLAERT AND M.S. OSORIO 1994. Statistical Disclosure in Two-Dimensional Tables: General Tables. JASA 89, 1547–57.

DOBRA, A. and S.E. FIENBERG 2000. Bounds for Cell Entries in Contingency Tables Given Marginal Totals and Decomposable Graphs. Contribution to a special set of Inaugural Articles by members of the National Academy of Sciences elected on April 27, 1999.

DUNCAN, G.T. AND S. FIENBERG 1999. Obtaining Information While Preserving Privacy: A Markov Perturbation Method for Tabular Data. In Statistical Data Protection (SDP '98) Proceedings, IDS Press.

DUNCAN, G.T. AND D. LAMBERT 1989. The Risk of Disclosure of Microdata. J. Bus Econ Stats 7, 207–217.

DUNCAN, G.T. AND S. MUKHERJEE 1991. Microdata Disclosure Limitation in Statistical Databases: Query Size and Random Sample Query Control. Symposium on Research in Security and Privacy, Oakland, California IEEE Computer Society, 278–287.

DUNCAN, G.T. AND S. MUKHERJEE 1994. Confidentiality Within Computer Databases. Statistica Applicata 6, 227-239.

DUNCAN, G.T. AND S. MUKHERJEE 2000. Get this title. *J. American Statistical Association*

DUTTA CHOWDHURY, S., G.T. DUNCAN, R. KRISHNAN, S. ROEHRIG AND S. MUKHERJEE 1999. Disclosure Detection in Multivariate Categorical Databases: Auditing Confidentiality Protection Through Two New Matrix Operators. To appear in Management Science.

EVANS, T., L. ZAYATZ AND J. SLANTA 1996. Using Noise for Disclosure Limitation of Establishment Tabular Data. U.S. Bureau of Census.

FELLEGI, I.P. 1972. On the Question of Statistical Confidentiality. JASA 67, 7–18.

GEURTS, J. 1992. Heuristics for Cell Suppression in Tables. Working paper, Netherlands Central Bureau of Statistics.

GLOVER, F. AND M. LAGUNA 1997. *Tabu Search*. Kluwer Academic Publishers, Boston, MA.

GOPAL, R.D., P.B. GOES AND R. GARFINKEL 1998. Interval Protection of Confidential Information in a Database. INFORMS JOC 10, 309–22.

KELLY, J.P. 1990. Confidentiality Protection in Two and Three-Dimensional Tables". Ph.D. Dissertation. Mathematical Sciences, University of Maryland, College Park, MD.

KELLY, J.P., B.L. GOLDEN AND A.A. ASSAD 1990. Using Simulated Annealing to Solve Controlled Rounding Problems. ORSA JOC 2, 174–185.

KELLY, J.P., B.L. GOLDEN AND A.A. ASSAD 1992. Cell Suppression: Disclosure Protection for Sensitive Tabular Data. Networks 22, 397-417.

KELLY, J.P., B.L. GOLDEN, A.A. ASSAD AND E.K. BAKER 1990. Controlled Rounding of Tabular Data. Opns. Res. 38, 760-772.

LOUGEE-HEIMER, R. 1989. Guaranteeing Confidentiality: The Protection of Tabular Data. Masters Thesis, Department of Mathematical Sciences, Clemson University.

MURALIDHAR, K., D. BATRA AND P.J. KIRS 1995. Accessibility, Security, and Accuracy in Statistical Databases: The Case for the Multiplicative Fixed Data Perturbation Approach. Man. Sci. 41, 1549-1564.

NARGUNDKAR, M.S. AND W. SAVELAND 1972. Random-rounding of tables to prevent statistical disclosures. Proc. Am. Stat. Assoc., 382-385

SULLIVAN, C.M. AND E. ROWE 1992. A Data Structure and Linear Programming Technique to Facilitate Cell Suppression Strategies. Proceedings of the Section on Survey Research Methods, American Statistical Association, Washington, D.C..

SULLIVAN, C.M. AND L. ZAYATZ 1991. A Network Flow Disclosure Avoidance System Applied to the Census of Agriculture. Proceedings of the Section on Survey Research Methods, American Statistical Association, Washington, DC, 363-368.

WILLENBERG, L.C.R.J., A.G. DE WAAL AND W.J. KELLER 1996. Some Methodological Issues in Statistical Disclosure Control. Fundacao Instituto Brasileiro de Geographica e Estatistica.

ZAYATZ, L. 1992. Using Linear Programming Methodology for Disclosure Avoidance Purposes I. Research Report, Statistical Research Division, Bureau of the Census, Washington, D.C.,

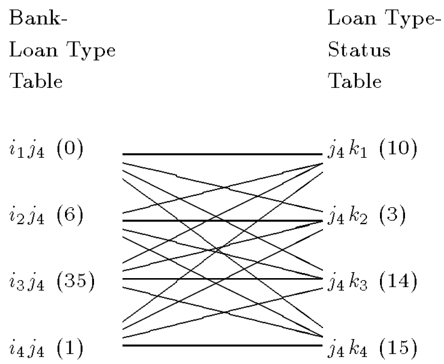

Figure 3: Network Structure for $j = 4$ in the Banking Example