Auditing Disclosure in Multi-Way Tables With Cell Suppression: Simplex and Shuttle Solutions

Stephen F. Roehrig
The Heinz School of Public Policy and Management
Carnegie Mellon University
Pittsburgh, PA 15213
roehrig@andrew.cmu.edu

Abstract: Statistical agencies typically collect sensitive, multi-dimensional data about individuals and organizations, but to protect respondants' confidentiality, publish only summaries in the form of lower-dimensional margins. If these tables still allow inferential disclosure, cell suppression may be used. In this talk we discuss disclosure auditing procedures that can be used to check the efficacy of such suppressions. In particular, we develop a close connection between a linear programming solution and Gröbner bases. Then we show how the dual of the LP approach is related to the shuttle algorithm of Buzzigoli and Giusti, and how both may be modified to accomodate suppressions.

1 Introduction

Consider the problem where the three 2-dimensional margins $A[i,j]$, $B[j,k]$ and $C[i,k]$ of a sensitive 3-dimensional table $T = [x_{ijk}]$, $i = 1,\ldots,I$, $j = 1,\ldots,J$, $k = 1,\ldots,K$ are proposed for release. Since T is sensitive, it is obviously of interest to be able to calculate tight upper and lower bounds on the values of each x_{ijk}. If the upper-lower bound pair for an x_{ijk}, often called its *feasibility interval*, is too revealing of the true value, suppression of marginals in the releasd tables might be considered as a protection mechanism. Thus it might be useful to know precisely how the inferred cell bounds are influenced by such suppressions. In this paper, we look at several approaches for generating bounds, and consider the extent to which these solution techniques inform the process of suppression selection. We assume the sensitive data x_{ijk}, and thus the margins A, B and C, are non-negative integers.

A randomly generated $3\times3\times3$ example that we use for illustration is shown in Figure 1. The three 3×3 tables above the horizontal line are the three levels ($k = 1,2,3$) of T, while

$k = 1$

			7
			6
			4
7	4	6	17

$k = 2$

			8
			5
			5
6	5	7	18

$k = 3$

			4
			5
			45
5	5	44	54

6	5	8	19
6	5	5	16
6	4	44	45
18	14	57	89

Figure 1: The Basic Problem

the table below the line is the marginal sum over k (i.e., $A[ij]$). The problem faced by a data intruder is to find bounds on the sensitive, unpublished interior cells in the microdata tables above the horizontal line, given only the marginal values provided. If these bounds are too tight, the data disseminator may consider disclosure-reduction measures, such as cell suppression.

2 The Simplex Method and Gröbner Bases

An obvious method for obtaining the feasibility interval for an unknown x_{ijk} is linear programming. The known $(n-1)$-way margins are used as equality constraints on the interior cells, and by sequentially maximizing and minimizing each unknown, all feasibility intervals are obtained. A number of variations on this approach are possible. First, because in the simplex algorithm an extremal solution for any variable is immediately apparent from the simplex tableau, this algorithm can be augmented by a step that looks, after each pivot, for any incidental upper or lower bounds. If discovered, this obviates the need for a seperate, dedicated search. Calculating bounds by starting the simplex over for each variable is unnecessarily wasteful. Second, if one is only concerned about feasibility intervals of width zero (i.e., exact disclosure), the algorithm need only look for two feasible solutions with different values of the unknown. For the simplex method, a single non-degenrate pivot provides proof against exact dislosure, and indeed the pivot calculation needn't even be done. Third, for two dimensional tables, a very efficient network algorithm (see Gusfield [11]) can replace linear programming.

In three and higher dimensions, there appears to be no adequate network representation for the bounding problem, so other approaches have been formulated to overcome the perceived complexity of linear programming. We examine one of these (the "shuttle" algorithm [1]) in a later section. In this section, we look at a fairly new algebraic approach to integer programming solutions to the bounding problem.

Diaconis and Sturmfels [7] considered the problem of sampling from the set of all n-way tables having fixed margins of various orders. In particular, they showed that this problem can be solved by exploiting a basis of a certain polynomial ideal that captures the additive relationships between cell values and the marginals they contribute to. The members of this basis are interpreted as "moves" or modifications to the contents of the n-way table that preserve the marginal values. For a 2-way table with known margins, a typical move is of the form $\begin{bmatrix} + & - \\ - & + \end{bmatrix}$, which alters four table entries, leaving the remainder of the table unchanged. For a 3-way table such a move might look like $\begin{bmatrix} + & - \\ - & + \end{bmatrix} \begin{bmatrix} - & + \\ + & - \end{bmatrix}$ where modifications are made over rows, columns and levels. This approach assumes that some starting point, a feasible instantiation of the table agreeing with the margins, is known. For a statistical agency publishing the data they collect, such a starting point is obviously available.

Diaconis and Sturmfels were interested primarily in exploring the space of tables with given margins. Elements of the basis are drawn randomly and executed, resulting in a random walk through the space. Thus their approach didn't focus on finding bounds for individual cell entries, although they proved that any table exhibiting an extremal value for a cell could be reached by a finite sequence of basis moves. Thomas [13] and Tayur, Thomas and Natraj [12], extending work of Conti and Traverso [4], showed how a Gröbner basis for an extension ideal provided the "directionality" needed to move monotonically to an upper or lower bound for any cell value. A set of moves with this property is called a *test set* in the integer programming literature (see, e.g., Weismantel [14] for a comparison of known test sets).

In this section we sketch out a correspondence between the basis moves of Diaconis and Sturmfels and pivots in the simplex solution of a linear programming formulation of the bounding problem. We then illustrate how the basis changes when some of the marginal values are unknown (as might be the case under a cell suppression protection scheme).

The constraint set for the 3-way problem is

$$\sum_{j=1}^{K} x_{ijk} \quad = \quad A[i,j] \qquad (1)$$

$$\sum_{i=1}^{I} x_{ijk} = B[j,k] \quad (2)$$

$$\sum_{j=1}^{J} x_{ijk} = C[ik] \quad (3)$$

For the $3 \times 3 \times 3$ example, there are $IJK = 27$ variables and $IJ + JK + IK = 27$ constraints. There are only $(I-1)(J-1)(K-1) = 8$ degrees of freedom, so upon row reduction we should find eight variables which, if given concrete values, will determine the rest. After row reduction, we get Figure 2. The variables are numbered with the correspondence

$$\begin{array}{cccccc} 111 & 112 & 113 & 121 & \ldots & 333 \\ 1 & 2 & 3 & 4 & \ldots & 27 \end{array}$$

Note that the columns with more than one 1 ($x_{222}, x_{223}, x_{232}, x_{233}, x_{322}, x_{323}, x_{332}, x_{333}$) form a non-basic set for an LP. While at this stage (i.e., before a phase one simplex procedure), setting these variables to zero doesn't yield a feasible solution, it is basic.

After applying phase 1 of the simplex method, the tableau of Figure 3 results. This tableau illustrates an important structure. Each of the non-basic columns represents a possible pivot, a "move," that is a way of altering the current values of the x_{ijk} in such a way that the marginals are fixed. These moves can be visualized by the "+/-" tables of Diaconis and Sturmfels. For example, the `x1` column, corresponding to x_{111}, is just the following move (x_{111} is considered a "+", since if it enters the basis it will increase from its current value of zero).

$$\begin{array}{ccc} + & - & 0 \\ - & + & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} - & + & 0 \\ + & - & 0 \\ 0 & 0 & 0 \end{array}$$

A more complicated move is that of `x25` (x_{331}):

$$\begin{array}{ccc} 0 & + & - \\ + & - & 0 \\ - & 0 & + \end{array} \quad \begin{array}{ccc} - & 0 & + \\ + & 0 & - \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} + & - & 0 \\ -2 & + & + \\ + & 0 & - \end{array}$$

Before proceeding further with the simplex tableau, here are some observations about the set of possible moves. Fixing the three 2-dimensional margins, the set S of ways of filling out T consistent with those margins is a subset of $\mathbf{R}^{IJK}$, or for our example, a subset of $\mathbf{R}^{27}$. Consider the map $A : \mathbf{R}^{IJK} \to \mathbf{R}^{IJ+IK+JK}$ which takes an instantiation of T to its

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25	x26	x27	b
y1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	1	1	55
y2	0	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	0	0	-1	0	-4
y3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	-1	0	0	0	0	0	-1	0	0	-1	-45
y4	0	0	0	1	0	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0	0	0	0	-1	-1	0	0	0	-5
y5	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	5
y6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	5
y7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0	0	0	0	-1	-1	-43
y8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	7
y9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	44
y10	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	-1	0	-1	-1	0	0	0	0	0	0	0	0	0	-4
y11	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	5
y12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	5
y13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	5
y14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	5
y15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	-1	-1	0	-1	-1	-44
y16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	5
y17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	45
y18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	4
y19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	44

Figure 2: Row-Reduced Equations

		x 1	x 9	x22	x16	x14	x24	x25	x26
x 3	1.0	-1.0	-1.0	1.0	1.0	-1.0	1.0	1.0	0.0
x23	4.0	0.0	0.0	-1.0	0.0	0.0	-1.0	0.0	0.0
x 4	1.0	-1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
x15	2.0	-1.0	0.0	1.0	1.0	-1.0	0.0	1.0	0.0
x 5	1.0	0.0	0.0	1.0	0.0	-1.0	1.0	0.0	0.0
x 2	5.0	0.0	1.0	-1.0	-1.0	1.0	-1.0	-1.0	0.0
x10	3.0	-1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
x 8	2.0	0.0	-1.0	0.0	1.0	0.0	0.0	1.0	0.0
x11	0.0	0.0	-1.0	0.0	1.0	-1.0	0.0	1.0	1.0
x17	5.0	0.0	1.0	0.0	-1.0	0.0	0.0	-1.0	-1.0
x 7	6.0	0.0	0.0	0.0	-1.0	0.0	0.0	-1.0	0.0
x18	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	1.0	1.0
x13	3.0	1.0	0.0	-1.0	-1.0	0.0	0.0	-1.0	0.0
x 6	3.0	1.0	0.0	-1.0	-1.0	1.0	-1.0	-1.0	0.0
x27	44.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	-1.0
x20	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	-1.0
x21	1.0	0.0	0.0	0.0	0.0	0.0	-1.0	1.0	1.0
x19	4.0	0.0	0.0	-1.0	0.0	0.0	0.0	-1.0	0.0
x12	3.0	1.0	1.0	-1.0	-1.0	1.0	0.0	-2.0	-1.0
z	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Number of phase 1 pivots = 22

Figure 3: Simplex Tableau After Phase 1

margins. The definition of this map for our example is just the contraint set (Equations [1–3]). We want to characterize those vectors x such that $Ax = b$, where b is the known vector of margins. In general, this set is a hyperplane in $\mathbf{R}^{IJK}$, but since we are only interested in non-negative solutions, S is the intersection of this hyperplane with the non-negative orthant $\{x \in \mathbf{R}^{IJK} : x_i \geq 0, i = 1, \ldots, IJK\}$.

We are interested in the null space $K = \{x : A(x) = 0\}$ of this linear map, since if $x, y \in S$, $x - y \in K$. That is, if we know any point $x \in S$, every other point in S has the form $x + \eta$, where $\eta \in K$. In particular, some move in K takes a known $x \in S$ to a point having the maximum or minimum of x_{ijk}. The dimension of K is just the number of degrees of freedom, for fixed margins, in assigning values to T. This is $(I-1)(J-1)(K-1)$.

Now consider the set of $2 \times 2 \times 2$ moves that increase, say, x_{111}. One such move, shown earlier, is

$$\begin{array}{ccc} + & - & 0 \\ - & + & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array} \quad \begin{array}{ccc} - & + & 0 \\ + & - & 0 \\ 0 & 0 & 0 \end{array}$$

Placing a "+" in cell (1,1,1), there are four other locations for the second "+" in level $k = 1$ (they are (2,2,1), (2,3,1), (3,2,1) and (3,3,1)). Having chosen one, the complementary "-" locations in level $k = 1$ are determined. For each such choice at $k = 1$, there are two other levels in which the complementary -/+ square can be placed, giving a total of eight simple moves. In general, there are $(I-1)(J-1)(K-1)$ moves of this type, and each is in K. If it can be shown that these moves are linearly independent, then we have a basis for K.

To prove linear independence, there is an obvious general extension to the following argument for the $3 \times 3 \times 3$ case. Written as vectors, the moves are

$$\begin{array}{ccccccccccccccccccccccccccc}
+ & - & 0 & - & + & 0 & 0 & 0 & 0 & - & + & 0 & + & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & - & 0 & 0 & 0 & 0 & - & + & 0 & - & + & 0 & 0 & 0 & 0 & + & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & 0 & - & - & 0 & + & 0 & 0 & 0 & - & 0 & + & + & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & 0 & - & 0 & 0 & 0 & - & 0 & + & - & 0 & + & 0 & 0 & 0 & + & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
+ & - & 0 & - & + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & + & 0 & + & - & 0 & 0 & 0 & 0 \\
+ & - & 0 & 0 & 0 & 0 & - & + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & + & 0 & 0 & 0 & 0 & + & - & 0 \\
+ & 0 & - & - & 0 & + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & + & + & 0 & - & 0 & 0 & 0 \\
+ & 0 & - & 0 & 0 & 0 & - & 0 & + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & + & 0 & 0 & 0 & + & 0 & -
\end{array}$$

A basic result from linear algebra says that the number of independent rows in this set—its row rank—equals the column rank. There are 8 columns with a single "-" entry, and these occur in different rows, so these 8 column vectors are linearly independent. Thus the column rank (and so the row rank) is 8, showing that these 8 moves are linearly independent.

In general, then, there are $(I-1)(J-1)(K-1)$ basic ways of modifying the cell entries of T that leave the margins fixed. Linear combinations of these moves also leave the margins fixed, but these are all there are. Naturally, many linear combinations will violate the non-negativity requirement on entries of T, and so cannot be used directly. The standard simplex method provides a way to organize the legitimate moves, and suggests those moves that lead iteratively to upper or lower bounds of any cell entry of T.

To illustrate, suppose `x3` is to be maximized. Referring to Figure 3 again, the first step is to replace the z row with `x3`'s row. Then note that `x3` can be increased by performing any of the moves given by columns `x22`, `x16`, `x24` or `x25`, because their z row entries are positive. Choosing `x22` arbitrarily, both `x12` and `x13` are candidates to leave the basis. Choosing `x12` and performing the move (pivot), we end up with the tableau of Figure 4. The locations of the non-basic variables within the 3-way table are shown in Figure 5. Note that the knowledge that these variables are zero makes it trivial to determine all the others, using only subtraction from the known marginals.

The non-pivot columns in the tableau are updated in the following way. If a coefficient in the outgoing variable row is non-zero, say in column j, that column has added to it some multiple of the pivot column. The size of the multiple is given, in the usual way, by the amount of increase possible until the outgoing variable just drops to zero. Because the jth column started as a move (i.e., in the null space K), and is changed only by having a multiple of another move added to it, it remains a move.

There are parallels here with the network simplex method. In the network simplex, substantial savings in computation accrue to the fact that every basis corresponds to a spanning tree, and the fact that a pivot amounts to the addition of an arc (forming a cycle) and the deletion of an arc (which breaks the cycle). At any stage, knowledge of the basis variables alone (the arcs in the current spanning tree) are sufficient to determine all the arc values in an especially simple way, requiring only additions and subtractions. The result is that updates are fast, in part because no multiplications/divisions need be done, and in part because efficient data structures have been found to organize the pivoting operation. Another byproduct is that because of the "spanning tree property of network bases" [3], optimal solutions (and all intermediate ones) are integer.

For our problem, the ordinary simplex basis completely determines of the all variables. Typically this is extremely easy, as in Figure 5, where all variable values can be read off immediately from the positions of the non-basics. Whenever this is the case, all values are integer, by arguing along the lines of the pure network case (all variables are determined by additions/subtractions of integers). There are exceptions, though, as illustrated in Figures 6 and 7. Here a fractional solution is found, albeit one which is intermediate in the process of maximizing `x1` (the maximum for `x1` is 6, an integer, and is found after several more pivots). Looking at the nonbasis elements, it can be seen that only 3 of the 19 basic variables are determined by simple subtractions. This leaves some leeway for the fractional values.

		x 1	x 9	x12	x16	x14	x24	x25	x26
x 3	4.0	0.0	0.0	-1.0	0.0	0.0	1.0	-1.0	-1.0
x23	1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	2.0	1.0
x 4	1.0	-1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
x15	5.0	0.0	1.0	-1.0	0.0	0.0	0.0	-1.0	-1.0
x 5	4.0	1.0	1.0	-1.0	-1.0	0.0	1.0	-2.0	-1.0
x 2	2.0	-1.0	0.0	1.0	0.0	0.0	-1.0	1.0	1.0
x10	6.0	0.0	1.0	-1.0	-1.0	1.0	0.0	-1.0	-1.0
x 8	2.0	0.0	-1.0	0.0	1.0	0.0	0.0	1.0	0.0
x11	0.0	0.0	-1.0	0.0	1.0	-1.0	0.0	1.0	1.0
x17	5.0	0.0	1.0	0.0	-1.0	0.0	0.0	-1.0	-1.0
x 7	6.0	0.0	0.0	0.0	-1.0	0.0	0.0	-1.0	0.0
x18	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	1.0	1.0
x13	0.0	0.0	-1.0	1.0	0.0	-1.0	0.0	1.0	1.0
x 6	0.0	0.0	-1.0	1.0	0.0	0.0	-1.0	1.0	1.0
x27	44.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	-1.0
x20	4.0	1.0	1.0	-1.0	-1.0	1.0	1.0	-2.0	-2.0
x21	1.0	0.0	0.0	0.0	0.0	0.0	-1.0	1.0	1.0
x19	1.0	-1.0	-1.0	1.0	1.0	-1.0	0.0	1.0	1.0
x22	3.0	1.0	1.0	-1.0	-1.0	1.0	0.0	-2.0	-1.0
z	4.0	0.0	0.0	-1.0	0.0	0.0	1.0	-1.0	-1.0

Figure 4: Simplex Tableau After One Pivot

$k = 1$

*			7
		*	6
		*	4
7	4	6	17

$k = 2$

			8
	*		5
		*	5
6	5	7	18

$k = 3$

		*	4
*			5
	*		45
5	5	44	54

6	5	8	19
6	5	5	16
6	4	44	45
18	14	57	89

Figure 5: The Non-Basics After One Pivot

The cause of the problem is the use of a pivot column containing a 2 (in this case, in the `x23` column). These are what Diaconis and Sturmfels call "moves of degree 9." They are a part of the Gröbner basis for the $3 \times 3 \times 3$ problem, although they are *not* part of the minimal generating set for the polynomial ideal. This suggests that although they may show up in the tableau, they are not necessary for finding all *integer* solutions. However, as Thomas [13] points out, the full Gröbner basis of a more complicated extension ideal is required for an integer programming test set.

Figure 8 gives an example where the simplex routine finds a fractional upper bound of 15.5 for x_{344}. This shows that the 3-way bounding problem, when the three $(n-1)$-way margins are given, is not inherently integer. It is well-known that the bounding problem for 2-way tables given their one-dimensional margins results in integer solutions, as the constraint matrix is totally unimodular. For the problem studied in Chowdhury et al. [2], where bounds on one $(n-1)$-way margin are obtained from the two remaining $(n-1)$-way margins, bounds are also guaranteed to be integer, because the problem decomposes into two-dimensional subproblems.

The example given above demonstrates that to obtain the tightest *integer* bounds for our problem, some form of integer programming must be applied. The Gröbner basis method of Thomas [13] seems well suited to the problem, provided the basis can be computed for problems of reasonable size. Experience to date suggests that this may be difficult. The fastest general purpose computer program for computing Gröbner bases, Macaulay (Grayson and Stillman, `http://math.uiuc.edu/Macaulay2`), consumes some 40 MB of memory, and requires many hours, to find the basis for a $3 \times 3 \times 3$ problem. We are not aware of any basis solutions for larger problems, although special-purpose programs might be written to make this possible. Of course, once the basis is determined, it can be used for any problem of the same dimensions, and an efficient implementation of the actual bounding procedure is not difficult, given the basis.

3 The Shuttle Algorithm

The shuttle algorithm of Buzzigoli and Giusti [1] is a clever iterative technique for finding upper and lower bounds on cells in n-way tables when all n margins of dimension $n-1$ are given. The method is related to the MCA matrix operators given in [2], and shares with the Frèchet/Bonferroni bound method of Fienberg [9] the fact that is starts with loose bounds and tries to tighten them. This is in contrast to simplex techniques that successively sharpen estimates from within the feasibility interval. Thus we might say that the shuttle algorithm works "from the outside in" while simplex works "from the inside out."

		x23	x18	x 4	x 2	x14	x24	x25	x26
x 3	4.0	-1.0	1.0	1.0	0.0	-1.0	0.0	-1.0	-1.0
x 1	2.0	1.0	-1.0	-1.0	-1.0	1.0	0.0	1.0	1.0
x12	0.0	1.0	-1.0	-1.0	0.0	1.0	1.0	0.0	0.0
x15	5.0	-1.0	0.0	1.0	0.0	-1.0	-1.0	0.0	0.0
x 5	5.0	-1.0	0.0	0.0	0.0	-1.0	0.0	0.0	0.0
x16	1.0	1.0	-1.0	0.0	-1.0	1.0	0.0	0.0	1.0
x10	5.0	-2.0	1.0	1.0	1.0	-1.0	-1.0	0.0	-1.0
x 8	3.0	1.0	0.0	0.0	-1.0	1.0	0.0	0.0	0.0
x11	1.0	1.0	0.0	0.0	-1.0	0.0	0.0	0.0	1.0
x17	4.0	-1.0	0.0	0.0	1.0	-1.0	0.0	0.0	-1.0
x 7	5.0	-1.0	1.0	0.0	1.0	-1.0	0.0	-1.0	-1.0
x 9	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	1.0	1.0
x13	0.0	1.0	0.0	-1.0	0.0	0.0	1.0	0.0	0.0
x 6	0.0	1.0	0.0	-1.0	0.0	1.0	0.0	0.0	0.0
x27	44.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	-1.0
x20	5.0	-1.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0
x21	1.0	0.0	0.0	0.0	0.0	0.0	-1.0	1.0	1.0
x19	0.0	1.0	0.0	0.0	0.0	0.0	1.0	-1.0	0.0
x22	4.0	-1.0	0.0	0.0	0.0	0.0	-1.0	0.0	0.0
z	2.0	1.0	-1.0	-1.0	-1.0	1.0	0.0	1.0	1.0

Pivoting on something other than -1

		x10	x18	x 4	x 2	x14	x24	x25	x26
x 3	1.5	0.5	0.5	0.5	-0.5	-0.5	0.5	-1.0	-0.5
x 1	4.5	-0.5	-0.5	-0.5	-0.5	0.5	-0.5	1.0	0.5
x12	2.5	-0.5	-0.5	-0.5	0.5	0.5	0.5	0.0	-0.5
x15	2.5	0.5	-0.5	0.5	-0.5	-0.5	-0.5	0.0	0.5
x 5	2.5	0.5	-0.5	-0.5	-0.5	-0.5	0.5	0.0	0.5
x16	3.5	-0.5	-0.5	0.5	-0.5	0.5	-0.5	0.0	0.5
x23	2.5	-0.5	0.5	0.5	0.5	-0.5	-0.5	0.0	-0.5
x 8	5.5	-0.5	0.5	0.5	-0.5	0.5	-0.5	0.0	-0.5
x11	3.5	-0.5	0.5	0.5	-0.5	-0.5	-0.5	0.0	0.5
x17	1.5	0.5	-0.5	-0.5	0.5	-0.5	0.5	0.0	-0.5
x 7	2.5	0.5	0.5	-0.5	0.5	-0.5	0.5	-1.0	-0.5
x 9	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	1.0	1.0
x13	2.5	-0.5	0.5	-0.5	0.5	-0.5	0.5	0.0	-0.5
x 6	2.5	-0.5	0.5	-0.5	0.5	0.5	-0.5	0.0	-0.5
x27	44.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	-1.0
x20	2.5	0.5	-0.5	-0.5	-0.5	0.5	0.5	0.0	-0.5
x21	1.0	0.0	0.0	0.0	0.0	0.0	-1.0	1.0	1.0
x19	2.5	-0.5	0.5	0.5	0.5	-0.5	0.5	-1.0	-0.5
x22	1.5	0.5	-0.5	-0.5	-0.5	0.5	-0.5	0.0	0.5
z	4.5	-0.5	-0.5	-0.5	-0.5	0.5	-0.5	1.0	0.5

Figure 6: A Pivot Leading to a Fractional Basis

$k = 1$

	*		7
*			6
		*	4
7	4	6	17

$k = 2$

*			8
	*		5
		*	5
6	5	7	18

$k = 3$

			4
		*	5
	*		45
5	5	44	54

6	5	8	19
6	5	5	16
6	4	44	45
18	14	57	89

Figure 7: Non-Basics Corresponding to a Fractional Solution

$k = 1$

3	2	1	1	7
1	0	40	2	43
50	2	2	3	57
1	1	1	1	4
55	5	44	7	111

$k = 2$

0	0	0	0	0
0	2	2	3	7
0	3	0	3	6
0	2	0	0	2
0	7	2	6	15

$k = 3$

0	2	1	44	47
0	2	2	1	5
1	3	1	49	54
58	2	0	2	62
59	9	4	96	168

$k = 4$

2	3	0	48	53
1	60	1	3	65
1	2	42	$\boxed{1}$	46
1	1	1	3	6
5	66	44	55	170

5	7	2	93	107
2	64	45	9	120
52	10	45	56	163
60	6	2	6	74
119	87	94	164	464

Figure 8: $4 \times 4 \times 4$ Problem With Fractional Upper Bound

$k = 1$

		5
		5
8	2	10

$k = 2$

		3
		3
4	2	6

5	3	8
7	1	8
12	4	16

Figure 9: Example $2 \times 2 \times 2$ Table

This observation suggests that there may be some connection between the shuttle algorithm and the dual of the simplex problem for finding bounds. After a brief review of the shuttle algorithm, we present the dual simplex problem and derive a close connection between the two.

Figure 9 shows a simple $2 \times 2 \times 2$ example that we use to explain the shuttle algorithm. The shuttle algorithm works by iteratively obtaining upper and lower bounds for each n-way table cell. At the zeroeth iteration, all lower bounds are taken at zero. Then from these lower bounds and the known margins, an initial set of upper bounds are found in the first iteration.

The 3-D table with the assumed zero lower bounds is shown at the top of Figure 10. To obtain the first upper bound for x_{111}, say, note that $x_{111} + x_{121} = 5$, and the assumed lower bound for x_{121} is zero. Thus a possible upper bound for x_{111} is $5 - 0 = 5$ (this generalizes in the obvious way for larger tables). Two other possible bounds are obtained in the same way from the other margins of which x_{111} is a part. Thus $8 - 0 = 8$ and $5 - 0 = 5$ are the other possibilities, and since we are looking for the smallest upper bound, we choose 5. The remaining cells in the table are processed similarly, resulting in the second set of tables in Figure 10.

Now lower bounds for each cell are determined from these new estimates for the upper bounds. To obtain the third set of tables in Figure 10 (the new lower bounds), the algorithm subtracts current upper bounds from margins, as follows. For x_{111} again, the margin x_{1+1} is 5, while the upper bound for x_{121} is 2. So a possible lower bound for x_{111} is $5-2 = 3$. Other possibilites are $8 - 5 = 3$ and $5 - 3 = 2$. Since we are after the largest lower bound, we take the maximum of the three, or 3. It is not hard to see that these bounds are legitimate, since at each step the calculation relies on legitimate bounds obtained earlier.

The next round of upper bounds are obtained as before from the current estimates of lower bounds. The algorithm "shuttles" back and forth between upper and lower bounds until no improvement is found. In our example, the upper bounds presented in the last set of tables in Figure 10 represent this convergent state.

The primal simplex approch to the bounding problem takes, as presented earlier, the margins as constraints and successively maximizes and minimizes each x_{ijk} individually. For the $2 \times 2 \times 2$ problem just presented, the formulation for the upper bound of x_{111} is as follows, where $x_{ijk} \geq 0$ for all i, j, k.

	x_{111}	x_{112}	x_{121}	x_{122}	x_{211}	x_{212}	x_{221}	x_{222}		
max $z =$	1	0	0	0	0	0	0	0		
s.t.	1	1	0	0	0	0	0	0	$=$	5 (x_{11+})
	0	0	1	1	0	0	0	0	$=$	3 (x_{12+})
	0	0	0	0	1	1	0	0	$=$	7 (x_{21+})
	0	0	0	0	0	0	1	1	$=$	1 (x_{22+})
	1	0	1	0	0	0	0	0	$=$	5 (x_{1+1})
	0	1	0	1	0	0	0	0	$=$	3 (x_{1+2})
	0	0	0	0	1	0	1	0	$=$	5 (x_{2+1})
	0	0	0	0	0	1	0	1	$=$	3 (x_{2+2})
	1	0	0	0	1	0	0	0	$=$	8 (x_{+11})
	0	1	0	0	0	1	0	0	$=$	4 (x_{+12})
	0	0	1	0	0	0	1	0	$=$	2 (x_{+21})
	0	0	0	1	0	0	0	1	$=$	2 (x_{+22})

The dual problem has variables y corresponding to each constraint in the primal. Since the primal constraints are equalities, the y's are unconstrained in sign. The dual problem that finds the maximum for x_{111} is the following.

	y_{11+}	y_{12+}	y_{21+}	y_{22+}	y_{1+1}	y_{1+2}	y_{2+1}	y_{2+2}	y_{+11}	y_{+12}	y_{+21}	y_{+22}		
min $w =$	5	3	7	1	5	3	5	3	8	4	2	2		
s.t.	1	0	0	0	1	0	0	0	1	0	0	0	$\geq$	1 (x_{111})
	1	0	0	0	0	1	0	0	0	1	0	0	$\geq$	0 (x_{112})
	0	1	0	0	1	0	0	0	0	0	1	0	$\geq$	0 (x_{121})
	0	1	0	0	0	1	0	0	0	0	0	1	$\geq$	0 (x_{122})
	0	0	1	0	0	0	1	0	1	0	0	0	$\geq$	0 (x_{211})
	0	0	1	0	0	0	0	1	0	1	0	0	$\geq$	0 (x_{212})
	0	0	0	1	0	0	1	0	0	0	1	0	$\geq$	0 (x_{221})
	0	0	0	1	0	0	0	1	0	0	0	1	$\geq$	0 (x_{222})

$k = 1$

0	0	5
0	0	5
8	2	10

$k = 2$

0	0	3
0	0	3
4	2	6

5	3	8
7	1	8
12	4	16

$k = 1$

5	2	5
5	1	5
8	2	10

$k = 2$

3	2	3
3	1	3
4	2	6

5	3	8
7	1	8
12	4	16

$k = 1$

3	1	5
4	0	5
8	2	10

$k = 2$

1	1	3
2	0	3
4	2	6

5	3	8
7	1	8
12	4	16

$k = 1$

4	2	5
5	1	5
8	2	10

$k = 2$

2	2	3
3	1	3
4	2	6

5	3	8
7	1	8
12	4	16

Figure 10: Steps of the Shuttle Algorithm

Any feasible solution to the dual problem represents an upper bound for x_{111}. A simple feasible dual solution can be obtained as follows. Of the columns for which there is a "1" in the first (x_{111}) row, choose the one with the smallest objective function coefficient and set the corresponding y variable to one. Set all other y to zero. For x_{111} we take $y_{1+1} = 1$. This clearly satisfies all the constraints, but need not be optimal. Note, however, this this procedure is identical to the first step of the shuttle algorithm. To get an upper bound for the other x_{ijk} variables, the "1" in the right-hand-side of the dual problem is substituted in turn in each row, so that for any problem, there is just one "1" and the rest "0". This leads to the same table of first-cut upper bounds that was obtained by the shuttle algorithm.

For lower bounds on the x_{ijk}, the dual problem is the same, except that the direction of the constraint inequalities is reversed, and the objective function should be maximized. The y variables are still unconstrained in sign, and once again there will be exactly one "1" in the right-hand-side, in the row for which a lower bound is desired. Note that an obvious feasible solution has $y = 0$ for all y. This corresponds to the choice of zero for the initial shuttle lower bounds. To get a sharper lower bound for x_{111}, we try to increase the objective function. Note, however, that increasing any y neccessitates a corresponding decrease in another, in order that the ≤ 0 constraints remain satisfied. We look for a pair of y variables that effect the largest increase in the objective. For x_{111}, such a pair is $y_{+11} = 1$, $y_{2+1} = -1$.

This choice of y's leads to an objective value of 3, and once again this corresponds to the result obtained by the shuttle algorithm. In fact, the same marginal values that were used in the shuttle to obtain this bound are the ones used in the dual simplex solution. x_{2+1} was used to obtain the upper bound on x_{211}, which was in turn used with x_{+11} to obtain the lower bound for x_{111}. Similar observations hold for the remaining lower bounds.

The final stage of the shuttle algorithm finds new upper bounds. The dual problem consequently reverts back to $\geq$ inequalities and minimization of the objective function. At this stage, three y variables are chosen to be nonzero, in such a way that the objective function is as small as possible. Two of the y's will be 1, and the third will be -1. The three obviously need to be chosen so that the constraints are satisfied. One choice is $y_{+11} = y_{22+} = 1$, $y_{2+1} = -1$, giving an objective value of 4. Another choice is $y_{1+1} = y_{22+} = 1$, $y_{+21} = -1$; a third is $y_{1+1} = y_{22+} = 1$, $y_{+22} = -1$. All of these correspond to combinations of marginals that were used by (or could have been used by, in the case of ties) the shuttle algorithm.

Buzzigoli and Giusti (1998) show that for a $2 \times 2 \times 2$ problem, the shuttle algorithm converges after the second upper bound step. There seems to be no obvious reason why additional shuttles might not produce better bounds for larger $I \times J \times K$ problems. In addition, *fractional* solutions are simply not considered. For the $4 \times 4 \times 4$ example of §2, the shuttle algorithm finds the upper bound for x_{344} to be 16, but clearly the best possible *integer* upper bound is 15 (there is in fact a feasible solution giving this value). This shows that the shuttle algorithm may not produce the tightest bounds. This reasoning applies as well to other techniques that approach upper (resp. lower) bounds from above (resp. below) by improving feasible dual integer solutions. The methods of Fienberg [9] appear to fall into this category.

Emperical evidence suggests that fractional solutions are extremely rare. Thus it seems worthwhile to examine the the shuttle algorithm further, with an eye towards ways in which it might suggest suppression patterns to prevent disclosure. Since the shuttle is similar in principle to the MCA operators in [2], and since those operators do give rise to effective suppression heuristics, we examine this possibility next.

4 The Shuttle Algorithm for Suppression Heuristics

If the upper and lower bounds on a cell in an n-way table are too narrow, this is considered a disclosure, and some protective action must then be taken. This might take the form of suppression of some entries in the released data. For our example, the n marginal tables of dimension $n-1$ are released, so these tables are the ones of interest. The optimal cell suppression problem is typically formulated as an integer program, having 0-1 variables that are indicators of suppressions. This program has constraints enforcing that the sharpest computable bounds (with the chosen suppressions in place) are no narrower than some prescribed limits.

Even for two dimensions this is a very difficult problem. In two dimensions, there are natural network analogues of the table, which allow the development of fast and often effective heuristics. A network representation in 3 and higher dimensions is possible, but it must be augmented with additional side constraints that, in general, make the usual network flow algorithms unusable. The recent work of Fischetti and Salazar [10] for two-dimensional tables is encouraging, and it appears possible to extend their formulation (and hopefully their results) to higher dimensions. Nonetheless, fast heuristics are worthwhile until exact methods become available.

Following on the work of Chowdhury et al. [2], Duncan et al. [8] developed cell suppression heuristics to protect a sensitive two-dimensional margin of a 3-D table when the two remaining two dimensional margins are published. Their approach relied on a disclosure auditing algorithm similar to the shuttle, but for their problem, guaranteed to find optimal bounds. We propose a similar formulation next.

Consider the marginals for a $3 \times 3 \times 3$ table shown in Figure 11. From these, it can be calculated (via the shuttle or LP) that the feasibility interval for x_{111} is $[5, 6]$. Assuming that this is too narrow for publication, we can use the shuttle to sketch a procedure for finding a

$k = 1$

			13
			2
			9
6	12	6	24

$k = 2$

			3
			15
			10
12	8	8	28

$k = 3$

			16
			8
			8
6	17	9	32

14	17	1	32
4	10	11	25
6	10	11	27
24	37	23	84

Figure 11: Sample Problem for Suppression Analysis

cell suppression pattern. Consider first extending the upper bound.

The upper bound of 6 for x_{111} is in this case produced on the first iteration of the shuttle, by subtracting the zero lower bounds for x_{211} and x_{311} from $x_{+11} = 6$. If x_{+11} were suppressed, then (ignoring for now the possibility of reconstructing x_{+11}) we might modify the shuttle algorithm to take the smaller of the remaining possibilities as the new upper bound. These are $x_{1+1} = 13$ and $x_{11+} = 14$. By this procedure, suppressing a margin may have the effect of increasing the (current) upper bound by some fixed increment. Note also that if *all* the margins for which x_{111} is an addend were suppressed, the derived upper bound would increase to infinity. (This again supposes that additional *complementary* suppressions are added to prevent the possiblity of recovering the proposed *primary* suppressions.)

To reduce the lower bound, a similar procedure is followed. The lower bound of 5 for x_{111} is obtained in the second shuttle iteration by subtracting upper bounds found at the first iteration. The largest of these is $x_{11+} - x_{112} - x_{113} = 14 - 3 - 6 = 5$. The subtractions producing the other two possible lower bounds both yield 0. It is possible to decrease the lower bound by finding a suppression that increases either the upper bound 3 for x_{112} or the upper bound 6 for x_{113}. Going back to the upper bounding iteration of the shuttle, it can be seen that suppressing $x_{1+2} = 3$ will drive the upper bound for x_{112} to 12. Similarly, suppressing x_{+13} drives the upper bound of x_{113} to 14. Note that choosing x_{+13} can serve two purposes: first, it decreases the lower bound for x_{111} as just described, and it also acts, at least partially, to protect the suppression at x_{1+1} chosen earlier.

The combination of the two suppressions just described succeeds in expanding the feasibility region of x_{111} to $[0, 13]$. There remains, however, the problem that one could still recover the suppressed values from the one-dimensional margins unless further action is taken. The complementary problem is, however, reduced to choosing suppressions in one or more (a maximum of three) two-dimensional tables, a familiar problem. For this example, having suppressed x_{1+1} and x_{+13}, complementary suppressions can be limited to the table $[x_{i+k}]$. We assume that techniques are in place to do this optimally or near-optimally.

The foregoing analysis suggests a simple greedy heuristic that might be suitable for tables with relatively few disclosures. For each disclosed cell in the n-way table, suppress 2-D marginal cells until the upper bound is driven above the required level. Do the same for the lower bound. Repeat this process for each disclosed cell, then choose complementary suppressions in one or more 2-D margin tables as necessary.

As mentioned earlier, a similar procedure was described in [8]. In addition to a simple greedy heuristic, a tabu search algorithm was given that used the information provided by an auditing algorithm much like the shuttle. The tabu search was shown empirically to be quite good. In all cases where an exact integer programming solution could be found, the tabu search found suppression patterns with identical numbers of suppressions. Furthermore, the tabu search greatly out-performed the gready heuristic.

5 Conclusions

In this paper we have considered three methods for auditing disclosure in n-way tables when the n marginal tables of dimension $n-1$ are released. The simplex method, with the enhancements mentioned in §2, is reasonably fast in practice, although it is substantially slower than the shuttle method discussed in §3. However, the shuttle can report bounds that are not the tightest possible in some circumstances. We do not know at present if the shuttle can be off by more than one, but if not, it may still prove useful in a situation where the required feasibility interval is wide.

The shuttle algorithm can also be used, as shown in §4, as the basis for suppression heuristics, as it exhibits a direct connection between suppressed marginal cells and the consequent bounds. Disclosure auditing via linear programming is also a natural technique to use when there are suppressions. By merely deleting some constraints, new bounds may be obtained. The integer programming test set provided by Gröbner bases can also be modified in a similar way to take into account suppressed cells. However, given the enormous number of possible suppression patterns, and the computational difficulty of finding Gröbner bases, this idea is currently unattractive. In any event, neither the linear programming algorithm nor the Gröbner basis test set give any particular insight into *which* cells to suppress if a disclosure exists.

Perhaps the most important contribution of the paper is a counterexample to the apparently widespread belief that the bounding problem in three dimensions will always result in integer solutions. For an auditing procedure that iteratively moves from looser constraints to tighter ones, this appears to pose a significant problem. Future work might profitably be spent characterizing those auditing problems that are—and are not—guaranteed to have integer solutions.

References

[1] 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 Statistical Data Protection '98*, March 1998, Lisbon.

[2] S.D. Chowdhury, G.T. Duncan, R. Krishnan, S.F. Roehrig and S. Mukherjee, "Disclosure Detection in Multivariate Categorical Databases: Auditing Confidentiality Protection Through Two New Matrix Operators", Working Paper #96-111, The Heinz School, Carnegie Mellon University, Pittsburgh, PA, 15213 (to appear in *Management Science*).

[3] Chvátal, V., *Linear Programming*, W.H. Freeman and Company, 1983.

[4] Conti, P. and C. Traverso, "Buchberger Algorithm and Integer Programming," in H.F. Mattson, T. Mora and T.R.N. Rao (eds.) *Applied Algebra, Algebraic Algorithms and Error Correcting Codes*, Lecture Notes in Computer Science 539, Springer-Verlag 1991, p. 130–139.

[5] L.H. Cox, "Some Remarks on Research Directions in Statistical Data Protection," in *Proceedings of Statistical Data Protection '98*, March 1998, Lisbon.

[6] L.H. Cox, "Network Models for Complementary Cell Suppression," *JASA* **90**, 1995, pp. 1453–62.

[7] Diaconis, P. and B. Sturmfels, "Algebraic Algorithms for Sampling from Conditional Distributions," *Annals of Statistics* **26**, no. 1, 1988, p. 363–397.

[8] G.T. Duncan, R. Krishnan, R. Padman, P. Reuther and S.F. Roehrig, "Cell Suppression to Limit Content-Based Disclosure," in *Proceedings of the Thirtieth Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, 1997.

[9] S.E. Fienberg, “Fréchet and Bonferroni Bounds for Multi-Way Tables of Counts With Applications to Disclosure Limitation,” Technical Report No. 691, February 1999, Department of Statistics, Carnegie Mellon University, Pittsburgh, PA 15213.

[10] M. Fischetti and J.J. Salazar, “Modeling and Solving the Cell Suppression Problem for Linearly-Constrained Tabular Data,” in *Proceedings of Statistical Data Protection ‘98*, March 1998, Lisbon.

[11] Gusfield, D. “A Graph Theoretic Approach to Statistical Data Security,” *SIAM J. Comput* **17**, no. 3, June 1988, p. 552–571.

[12] Tayur, S.R., R.R. Thomas and N.R. Natraj, “An Algebraic Geometry Algorithm for Scheduling in Presence of Setupa and Correlated Demands,” *Mathematical Programming* **69**, 1995, p.369–401.

[13] Thomas, R., “A Geometric Buchberger Algorithm for Integer Programing,” *Math of OR* **20**, no. 4, November 1995, p. 864–884.

[14] Weismantel, R., “Test Sets of Integer Programs,” *Mathematical Programming* **47**, 1998, p. 1–37.