Statistical Disclosure in Two-Dimensional Tables: General Tables

Filipa Duarte de Carvalho; Nico P. Dellaert; Margarida de Sanches Osorio

Journal of the American Statistical Association, Vol. 89, No. 428 (Dec., 1994), 1547-1557.

Stable URL:
http://links.jstor.org/sici?sici=0162-1459%28199412%2989%3A428%3C1547%3ASDITTG%3E2.0.CO%3B2-8

Journal of the American Statistical Association is currently published by American Statistical Association.

Statistical Disclosure in Two-Dimensional Tables: General Tables

Filipa Duarte de CARVALHO, Nico P. DELLAERT, and Margarida de Sanches OSÓRIO*

Confidentiality protection of data published in tables is a major problem for statistical offices. To obtain full cooperation of the respondents, it is required that information of individual respondents with a confidential character is kept from being disclosed. One method to avoid disclosure is the method of cell suppression, in which the values of a number of statistical cells are not published but are suppressed from publication. We discuss the method of cell suppression in general two-dimensional tables, in which row totals and column totals are always published. The values of the sensitive cells are replaced by a cross (×). Usually, additional suppressions are necessary to prevent the values of the sensitive cells from being calculated from the row or column totals. Because of these additional suppressions, useful information gets lost. We want to minimize the loss of information by making the best choice for the additional suppressions. Therefore, we introduce and compare the performance of some heuristics for solving this problem.

KEY WORDS: Cell suppression; Disclosure problems; General two-way tables.

1. INTRODUCTION

Statistical figures on economical and social aspects are published in many countries by the national statistical offices, for instance, the U.S. Census Bureau. Usually the figures are presented in tabulations, containing either frequency counts (often for sociological aspects) or magnitude data (for economical aspects). In the first case the values of the tables cannot take negative values. We will refer to this type of table as positive tables. In the second case it is possible that the values can take any real (or integer) value, for example, in the case of financial information, where positive entries could correspond to surplus values and negative entries could correspond to deficit values. We will refer to this type of table as general tables.

It is required that information of individual respondents with a confidential character be kept from publication. Several disclosure avoidance techniques can be used to achieve this, including aggregation, perturbation, random rounding, and cell suppression. (See Cox, Fagan, Greenberg, and Hemmig 1986 for a discussion of those techniques.) This article is concerned with one specific method, cell suppression, in which the values of a number of statistical cells are not published but are suppressed from publication.

The set of suppressed cells usually consists of two subsets: the set of sensitive cells, containing information that cannot be published according to a sensitivity criterion, and the set of complementary suppressions, which have to ensure that the published table is safe, implying that none of the values of sensitive cells can be derived from the published data. Cox (1976) mentioned some criteria for this sensitivity. For frequency counts, a cell that contains fewer than a prescribed threshold value may be considered as sensitive, because the probability that one respondent, with his further attributes, can be identified is too large. For magnitude data, dominance rules can be used, in which a cell can be defined as sensitive if the aggregate value of the corresponding attribute over n respondents (for small n) is at least $k\%$ of the cell value. This is referred to as the "n-respondent, $k\%$ rule" (see Cox 1978).

With every complementary suppression, useful information gets lost. The value of the information may vary from cell to cell. This can be modeled by giving the cell a so-called weight. Clearly, the suppression problem can now be described as follows. For a given table, with a given set of sensitive cells and given weight values for the other cells, we have to determine the set of complementary suppressions in such a way that the published table is safe, while the total loss of information (i.e., the total weight of the cells) is the minimum possible.

In this article we consider only the case of general tables; that is, tables in which the elements are allowed to take both positive and negative values. Suppression techniques in positive tables can be quite different (Kelly, Golden, and Assad 1992). The main difference is that in general tables, only the pattern of suppressed cells is important for the disclosure, whereas in positive tables, the values of the suppressed cells can also influence the disclosure. We first present methods to check whether disclosure in a published data table is possible. Some previous work has been done on this problem by Cox (1977, 1978, 1980) and others, including Fellegi (1972), Gusfield (1988), and Duncan and Lambert (1986). The second part of the article considers the problem of making a table safe with a minimum loss of information. The literature on this subject is rather poor, especially if the weight values of the cells can be different from each other. Kelly et al. (1992) have shown that the problem of positive tables is NP hard. Recently, DeSilets, Golden, Kumar, and Wang (1991) proposed a neural network model for the general problem. For this problem, we discuss the features of the optimal solution and present a number of heuristics based on a graph theoretic representation of the problem. We compare the heuristics with each other and, if possible, with the optimal solution, obtained using a simple branch-and-bound scheme.

This article is organized as follows. In Section 2 we give some definitions and show some results on the disclosure

* Filipa Duarte de Carvalho is a Member of Instituto Superior de Economia e Gestao, 1200 Lisboa, Portugal. Nico P. Dellaert is Associate Professor of Operations Research, Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, Netherlands. Margarida de Sanches Osório is a Consultant, 2570 Cascais, Portugal. This material is based on work supported by the Erasmus Exchange Program. The authors thank Gerard Hooghiemstra, Mike Keane, Mieke Bemelmans, Edwin Romeijn, Vitaly Strusevich, the late Albert Verbeek, two referees, and the associate editor for their useful and stimulating comments.

Journal of the American Statistical Association
December 1994, Vol. 89, No. 428, Theory and Methods

problem that can be used to construct heuristics. In Section 3 we present a heuristic for checking whether a given table is safe. We present the features of an optimal solution and the heuristics for making a table safe in Section 4, and in Section 5 we discuss a randomized algorithm which uses the heuristics introduced in Section 4. Finally, in Section 6 we present the results of the various heuristics for some randomly generated sets of test problems.

2. DEFINITIONS AND PROBLEM DESCRIPTION

We start this section by giving some definitions on the disclosure problem described in the introductory section. Next, we give an overview of the elements of graph theory that we will need in the remainder of the article. We end the section by giving some results characterizing safe tables.

2.1 The Disclosure Problem

In this section we define some standard terms pertaining to the disclosure problem.

Definition 2.1. A two-way table is a triple $(\mathbf{X}, \mathbf{a}, \mathbf{b})$, where $\mathbf{X}$ is a $(m \times n)$ matrix with (i, j)th element $x_{ij} \in \mathbb{R}$, $\mathbf{a}$ is a real $(m \times 1)$ vector whose ith element $\mathbf{a}_i$ contains the subtotal of row i, that is,

$$\mathbf{a}_i = \sum_{j=1}^{n} x_{ij},$$

and $\mathbf{b}$ is a real $(1 \times n)$ vector whose jth element $\mathbf{b}_j$ contains the subtotal of column j, that is,

$$\mathbf{b}_j = \sum_{i=1}^{m} x_{ij}.$$

For simplicity, throughout this article the term "table" will be used instead of "two-way table."

Definition 2.2. A *cell* of a table is a pair (i, j) with $1 \le i \le m$, $1 \le j \le n$. The set of all cells is denoted by

$$C \equiv \{(i, j): 1 \le i \le m, 1 \le j \le n\}.$$

Definition 2.3. A cell (i, j) of a table $(\mathbf{X}, \mathbf{a}, \mathbf{b})$ is a *sensitive cell* if its value may not be dissolved. The set of sensitive cells will be denoted by

$$S_1 \equiv \{(i, j) \in C : (i, j) \text{ is a sensitive cell}\}.$$

A table $(\mathbf{X}, \mathbf{a}, \mathbf{b})$ with corresponding set of sensitive cells S_1 will be denoted by $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$.

Definition 2.4. A cell (i, j) of a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ is a *complementary suppression* if it is hidden to avoid disclosure of sensitive cells. A table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ with corresponding set of complementary suppressions S_2 will be denoted by $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$.

Definition 2.5. A cell (i, j) of a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$ is a *suppressed cell* if it is a sensitive cell or if it is hidden to avoid disclosure of a sensitive cell, that is, $(i, j) \in S \equiv S_1 \cup S_2$.

Definition 2.6. A *completion* of a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$ is a set of values

$$D \equiv (d_{ij})_{(i,j) \in C}$$

for the cells in C, which is consistent with the given row and column subtotals, and such that $d_{ij} = x_{ij}$ for all unsuppressed cells $(i, j) \in C \setminus (S_1 \cup S_2)$.

Definition 2.7. *$\mathcal{K}$-disclosure* occurs in a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$ if, for some sensitive cell $(i, j) \in S_1$, the difference between the maximum and minimum value that cell can take, is less than $\mathcal{K}$, that is,

$$\max_D \{d_{ij} : D \text{ is a completion}\} - \min_D \{d_{ij} : D \text{ is a completion}\} < \mathcal{K}.$$

Definition 2.8. A table is *$\mathcal{K}$-safe* if $\mathcal{K}$-disclosure does not occur.

The first lemma of this article gives a characterization of a $\mathcal{K}$-safe table.

Lemma 2.9. A table if $\mathcal{K}$-safe if and only if for each sensitive cell (i, j) there exist two completions D' and D'' of the table such that $|d'_{ij} - d''_{ij}| \ge \mathcal{K}$.

The proof of this lemma follows directly from Definitions 2.7 and 2.8.

2.2 Graph Theory

The methods presented to solve the disclosure problem make use of graph theory. In Definition 2.1 we have defined a table as a triple $(\mathbf{X}, \mathbf{a}, \mathbf{b})$, where $\mathbf{X}$ is a real matrix and $\mathbf{a}$ and $\mathbf{b}$ are real vectors. A table, together with some subset E of its cells, can be presented by a bipartite graph $G = (V, E)$, where V is the set of vertices, $V = V_1 \cup V_2$, $V_1 = \{r_1, \ldots, r_m\}$ and $V_2 = \{c_1, \ldots, c_n\}$, where r_i corresponds to row i and c_j corresponds to column j. The set E of edges corresponds to a subset of the set C of cells of the table. In the remainder of this article we use the same symbol for a subset of the set C of cells and the corresponding set of edges, where the meaning will be clear from the context. For example, we can represent the table $(\mathbf{X}, \mathbf{a}, \mathbf{b})$ with its set of sensitive cells S by $G = (V, S)$. In Figures 1 and 2 we show an example of a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S)$ and its corresponding bipartite graph. After that we give a definition of a path and a circuit in a graph (see also Harary 1969).

-3	-1	3	2	1	2
4	X	2	-3	X	1
1	1	X	X	3	9
2	X	X	X	-2	4
0	X	4	1	X	2
4	1	3	8	2	18

*Figure 1. The Table (**X**, **a**, **b**, S).*

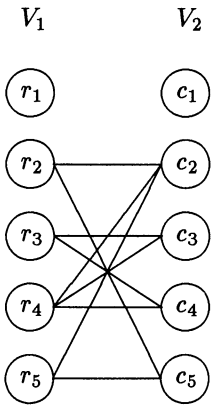

Figure 2. The Graph G = (V, S).

Definition 2.10. A *path* of length q is a sequence of q edges $P = \{u_1, \ldots, u_q\}$ with $u_1 = (i_0, i_1), \ldots, u_q = (i_{q-1}, i_q)$, where $i_j \in V$ for all $j = 0 \ldots q$ and $(i_{j-1}, i_j) \in E$ for all $j = 1 \ldots q$. Here i_0 is called the initial vertex of the path P, and i_q is called the terminal vertex of the path P. A *circuit* is a path whose initial and terminal vertices coincide.

Figure 2 shows two examples of a circuit, one containing the edges (r_3, c_3), (c_3, r_4), (r_4, c_4), and (c_4, r_3) and one containing the edges (r_2, c_2), (c_2, r_5), (r_5, c_5), and (c_5, r_2). Notice that the edge (r_4, c_2) does not belong to any circuit.

The next theorem (for which the proof is given in the Appendix) will give us a sufficient condition for a table to be $\mathcal{K}$-safe.

Theorem 2.11. A table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$, with $S_1, S_2 \subset C$, is $\mathcal{K}$-safe for every $\mathcal{K} > 0$ if every edge in the corresponding bipartite graph $G = (V, S_1 \cup S_2)$ (i.e., every suppressed cell) belongs to a circuit.

Now we would like to prove the necessity of this condition for a table to be $\mathcal{K}$-safe. Therefore, we first consider an arbitrary suppressed cell $(i_0, j_0) \in S$ and define the set $Y = S \backslash (i_0, j_0)$ and two collections of sets $\{A_k\}$ and $\{B_k\}$, $k = -1, 0, 1, 2, \ldots$, which are given by

$$A_{-1} = B_{-1} = \varnothing \quad \text{and} \quad A_0 = \{i_0\};$$

$$B_0 = \{j : (i_0, j) \in Y\},$$

and for $k \geq 1$,

$$A_k = \{i : j \in B_{k-1} : (i, j) \in Y\} \quad \text{and}$$

$$B_k = \{j : i \in A_k : (i, j) \in Y\}.$$

Then $A_k \subseteq A_{k+1}$ and $B_k \subseteq B_{k+1}$. Let $N = \min\{k \mid B_k = B_{k-1}\}$, implying that $A_k = A_{k+1}$ and $B_k = B_{k+1}$ for $k \geq N$. We call A_N the set of marked rows and B_N the set of marked columns. Both sets are finite, because the number of rows and columns is finite. Now we can state the following:

Theorem 2.12. For all elements $(i, j) \in Y$ holds that $i \in A_N$ if and only if $j \in B_N$.

Theorem 2.13. Consider an element $(i_0, j_0) \in S$ together with its set of marked rows A_N and its set of marked columns B_N. This element can be disclosed if and only if column j_0 does not belong to B_N.

The element (i_0, j_0) belongs to a circuit if and only if j_0 belongs to the marked columns. Therefore, we can now state the following result.

Result 2.14. An element $(i_0, j_0) \in S$ can be disclosed if and only if there is no circuit in $G = (V, S)$ containing this element.

An important result of these theorems is that the safety of a table in no way depends on the values x_{ij}, $\mathbf{a}_i$, or $\mathbf{b}_j$. This property is the main difference between general tables and positive tables. A related result of the theorem is that a table that is $\mathcal{K}$-safe for a specific $\mathcal{K}$ value is safe for any other value of $\mathcal{K}$. Therefore, in the remaining sections we will use the term "safe table" to denote $\mathcal{K}$-safe tables.

3. SAFETY CHECK

In this section we introduce an algorithm to determine the safety of a table that makes use of solution methods for the shortest-path problem.

3.1 Algorithm

As we have shown in Result 2.14, a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$ is safe if and only if for every sensitive cell in S_1 a circuit can be found containing that cell. We have also shown how a table with corresponding set of suppressed cells, $S = S_1 \cup S_2$, can be represented by an undirected bipartite graph, $G = (V, S)$. It is clear that finding a circuit containing a given edge (i, j) of a graph is equivalent to finding a path from i to j that does not contain (i, j). Therefore, we can check whether a circuit containing some cell (i, j) exists by deleting this cell from S, and then finding a path from i to j. Only if such a path exists is the cell corresponding to (i, j) safe. This is the basic idea behind the algorithm that we propose for checking whether a table is safe:

Checking.

Step 0. Set $T := S_1$.

Step 1. Choose some $(r_i, c_j) \in T$.
If (r_i, c_j) does not belong to a circuit in $G = (V, S)$, then STOP—the table is not safe.

Step 2. Let $P = \{(r_i, c_j), e_2, \ldots, e_q\}$ be a circuit containing (r_i, c_j).

Step 3. Set $T := T \backslash P$; that is, remove all cells in P from T.
If $T \neq \phi$, then go to Step 1.

Step 4. STOP—the table is safe.

Obviously, if we attach a certain "distance" or weight to each edge in S, and if there exists at least one path from i to j, then there also exists a shortest path between i and j. We will use this property for the disclosure problem. Because "distances" are nonnegative, and we need to find the shortest path between two vertices of the graph for each sensitive cell, we can use Dijkstra's shortest-path algorithm. This well-known algorithm finds shortest paths from a source vertex i to all other vertices, including j. The basic idea of the algorithm is to fan out from vertex i and label vertices in order of their distances to i. Each vertex v has a label, denoted by $d(v)$; the label is permanent once we know that it represents the shortest distance from i to v; otherwise, it is temporary.

Initially, we give vertex i a permanent label of zero and each other vertex k a label equal to w_{ik} if $(i, k) \in S$ and ∞ otherwise. At each iteration, the label of a node k is its shortest distance from the source vertex along a path whose internal vertices are all permanently labeled. The algorithm selects a vertex k with the minimum temporary label, makes it permanent, and scans edges in S from vertex k to update the distance labels of adjacent vertices. The algorithm terminates when it has designated all vertices as permanently labeled.

For the safety check problem, we set the distances of all edges in the set S equal to zero. In other words, in this case all *feasible* paths between a pair of vertices are also *shortest paths*. In the next section we use the same algorithm to make a table safe, with a minimum loss of information. But in that problem, we no longer restrict ourselves to the edges belonging to S, but we consider all edges in C.

Dijkstra's algorithm has complexity $O(|V|^2)$, where $|V|$ denotes the cardinality of the vertex set V. Note that $|V| = |V_1| + |V_2| = m + n$. The algorithm previously introduced will have complexity $O(|V|^2 \times |S_1|)$, where $|S_1|$ denotes the cardinality of the set S_1. If we do not impose a stopping condition before the set T is empty, then the algorithm will give us the set of unsafe sensitive cells, which will be empty in case the table is safe. Notice that in Step 3 all cells in P (at least the ones that also belong to T) are removed from T, to avoid some redundant work.

4. MAKING A TABLE SAFE

In this section we concern ourselves with the problem of making a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe. Suppose that we are given a positive weight w_{ij} for every cell (i, j) in $C \backslash S_1$ and a weight zero for every cell (i, j) in S_1. This weight corresponds to the information value that gets lost if the cell is suppressed from publication. We assume that the weight of each cell is fixed and not related to other suppressions. Special situations are equal weights for all cells, considered, for instance, by Cox (1980) and Gusfield (1988), and weights equal to the table value (Kelly et al. 1992). Our goal is then to find a set of complementary suppressions S_2 with minimum weight that will make a table safe when suppressed from the table together with the set of sensitive cells.

We first describe methods to find the optimal solution, then some heuristic methods. The heuristic methods we propose again are based on Result 2.14. Analogously to the previous section, we face the problem of finding paths in an undirected bipartite graph.

4.1 Optimal Solution

Figure 3 gives an example of a weight matrix. To make this table safe, we can determine a circuit with minimal weight for both of the primary suppressions (**X**) separately. The complementary suppressions for this solution are encircled in the table, and their total weight is 12. The complementary suppressions for the optimum solution are boxed. Now both primary suppressions are in the same circuit, and the total weight is 9.

From this example, it seems clear that if there are P primary suppressions, they can all be in the same circuit in the optimal solution, or partly in different circuits. This idea could be used to construct the optimal solution, by considering all combinations of subsets of primary suppressions and finding for each subset the best circuit containing all its elements. The number of combinations of subsets grows exponentially in the number of primary suppressions P. Therefore, this method is not suitable for larger P values.

②	3	5	10	②	4
[5]	X	6	8	10	③
7	[2]	[1]	7	9	2
7	5	1	1	1	4
10	①	8	10	2	②
X	9	[1]	5	②	4

Figure 3. An Example of a Weight Matrix.

An alternative method that we use in Section 6 makes use of the algorithm to check whether a table is safe as described in Section 3. We use a standard depth-first branch-and-bound algorithm. In each node we branch by choosing an unsuppressed cell that we either suppress or forbid. Of course we start in the rows and columns with only one suppression. Subtrees are pruned as soon as we find a safe table with a lower weight than the best solution so far. As we see in Section 6, this branch-and-bound algorithm cannot always be used for larger-sized tables, because of its exponential character.

Kelly et al. (1992) have shown that a special version of the problem with positive tables is NP hard. Until now, the complexity of the problem with general tables is an open question and a very interesting research topic. Because a polynomial algorithm for this problem has yet to be found, we present a number of heuristics that can give good results within reasonable time.

4.2 Heuristics

The class of heuristics that we propose for making a table safe is based on the problem of finding paths in the graph $G' = (V, C)$ with given nonnegative weight values for all cells. This graph is undirected, bipartite, and complete. The set V corresponds to $V_1 \cup V_2$, where $V_1 = \{r_1 \ldots r_m\}$ and $V_2 = \{c_1 \ldots c_n\}$. For every edge $(r_i, c_j) \in S_1$ (the set of sensitive cells), we need again to find a circuit containing that edge. Like before, we delete (r_i, c_j) from G' and then find a path from r_i to c_j. Because we are working with the complete graph G' for this problem, such a path will always exist. The heuristic we propose for this problem is as follows:

Circuit-finding Heuristic.

Step 0. Set $T := S_1$.
Set $S_2 := \phi$.
Step 1. Choose some $(r_i, c_j) \in T$.
Step 2. Find a circuit containing (r_i, c_j) in the graph G'.
Denote this circuit by $P = \{(r_i, c_j), e_2, \ldots, e_q\}$.
Step 3. Set $T := T \backslash P$; that is, all sensitive cells in P are removed from T.

Set $S_2 := S_2 \cup (P \backslash S_1)$; that is, all nonsensitive cells in P are added to S_2.

Step 4. If $T \neq \phi$, go to Step 1.
Otherwise, STOP—S_2 is a set of complementary suppressions that makes the table safe.

The weight of the solution obtained with this heuristic is the sum of the weights w_{ij} for all cells $(i, j) \in S_2$. Note that the table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ is safe if the heuristic terminates with $S_2 = \phi$. But at this point we cannot guarantee that the set of complementary suppressions S_2 found by the heuristic will always be empty if the table is safe. This depends on the algorithm that we use in Step 2. Of course, we can always perform the checking algorithm first to guarantee an empty set S_2 for safe tables.

The foregoing heuristic can be implemented in several ways. In Step 2 we need to find a circuit in the graph that contains some specific edge. One possibility is to use Dijkstra's algorithm in the way just described. Therefore, we adapt the original weight ("distance") matrix. The weight matrix used by Dijkstra, say $\bar{\mathbf{W}} = (\bar{w}_{ij})$, is constructed as follows. Initially, for all sensitive cells $(i, j) \in S_1$, we set $\bar{w}_{ij}$ to some nonpositive value, say $\bar{w}_S$, while the other cells (i, j) have weight $\bar{w}_{ij} = w_{ij}$. If $\bar{w}_S$ is chosen equal to zero, then this algorithm will find the *shortest* path among all feasible paths. Due to the properties of Dijkstra's algorithm, the heuristic will return with an empty set of complementary suppressions if the original table is safe. If some of the weights of the sensitive cells are initially set to a negative value, then finding the shortest path may be NP hard if circuits with a negative total weight can occur (Garey and Johnson 1979). Because of the speed, we will still use Dijkstra's algorithm to find a good path (though not always the shortest possible path) which uses each vertex at most once.

The reason for setting the weights corresponding to sensitive cells to a nonpositive value $\bar{w}_S$ is the fact that those cells will then have a smaller weight than all other cells of the table, and consequently they will be a more attractive choice than the other cells in the path-finding algorithms just described. Not only is no extra weight associated with the inclusion of a sensitive cell in a circuit, but also a new path for this sensitive cell does not need to be found. For this reason, negative values for $\bar{w}_S$ may give better results (i.e., a smaller sum of weights of suppressed cells) than a value of zero. Sensitive cells that are taken out of the set T no longer have negative weight values, but always a value of zero. There is no unique best value for $\bar{w}_S$. For example, the weight matrix in Figure 3 contains two sensitive cells that can be made safe by two different circuits of weight 6, or by one single circuit of weight 9 (of course, containing both sensitive cells). For a value $\bar{w}_S > 3$, we may find this optimal solution. Notice, however, that for $\bar{w}_S > 6$, we would also prefer the single circuit, even if the weight would be more than 12. Generally, $\bar{w}_S$ can be small if we consider large tables or tables with many sensitive cells.

Whenever new complementary suppressions are added to set S_2, the weights $(\bar{w}_{ij})$ of the cells in this set will be set to zero. This will make them more attractive in future path-finding steps, but still less (or equally attractive if $\bar{w}_S = 0$) attractive as the sensitive cells in the set T. The choice of the zero value is rather logical, because the weight of the set S_2 will not change if some cell in S_2 is chosen again in a circuit. If Dijkstra's algorithm is used for finding paths in G', then the heuristic will have complexity $O(|V|^2 \times |S_1|)$, where $|V|$ denotes the cardinality of the vertex set of the graph G' and $|S_1|$ denotes the cardinality of the set S_1.

Dellaert (1984) mentioned several other methods for finding a circuit in the graph G'. In the minimum rectangle method, for an edge $(r_k, c_l) \in S_1$, the algorithm finds the shortest circuit P containing the edge (r_k, c_l) and exactly three other edges; that is, P is of the following form:

$$P = \{(r_k, c_l), (c_l, r_q), (r_q, c_s), (c_s, r_k)\}.$$

The weight matrix $\bar{\mathbf{W}}$ used will be the same as described previously for Dijkstra's algorithm. Note that this method finds a path containing the least number of edges, because in any bipartite graph the minimum number of edges in a cycle is four. The advantage of this algorithm to solve the problem of finding paths in G' is the fact that the heuristic will have complexity $O(|S_1| \times m \times n)$, where m and n denote the number of rows and columns of the table. This complexity is considerably smaller than that when Dijkstra's algorithm is being used to find paths in G'.

The disadvantage of the minimum rectangle method is that it does not take into account the possible existence of paths with more than three edges having smaller weight. Therefore, we could expect the version of the heuristic using this algorithm generally to perform considerably worse than the version using Dijkstra's algorithm. In particular, this version of the heuristic may return with the set S_2 nonempty even in the case where the original table is safe.

In the next subsection we propose some improvement methods for the heuristic previously described.

4.3 Improvement Methods

With the heuristic described in the previous subsection, we obtain the set of edges S_2 of the graph G' that corresponds to the set of complementary suppressions of the table. But some of the edges in S_2 may not be necessary. The improvement methods described in this section will be used to identify redundant edges in S_2. The graph we consider for the improvement methods is $G'' = (V, S)$, where $S = S_1 \cup S_2$ is the set of suppressed cells.

The first improvement method is based on the fact that it is possible that an edge in S_2 has become redundant because of complementary suppressions that have been added later. It is clear that in a safe table, the number of suppressed cells in each row and column cannot be equal to one. Therefore we check whether a complementary suppression $(i, j) \in S_2$ is redundant only if there are more than two suppressed cells in row i and column j.

Delete-and-Check.

Step 0. Set $T := \{(i, j) \in S_2 : |\{k : (i, k) \in S\}| > 2$ and $|\{k : (k, j) \in S\}| > 2\}$.

Step 1. Choose some cell $(r_i, c_j) \in T$.
If the table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S \backslash (r_i, c_j))$ is safe, then $S_2 := S_2 \backslash \{(r_i, c_j)\}$.

Step 2. $T := T \setminus \{(r_i, c_j)\}$.
If $T \neq \phi$, then go to Step 1.

Step 3. STOP—S_2 is a new set of complementary suppressions that make the table safe.

In Step 1 of the foregoing improvement method, we can use the algorithm for checking table safety introduced in Section 3.1. As we have already mentioned, this algorithm has complexity $O(|V|^2 \times |S_1|)$. Therefore, the delete-and-check method will have complexity $O(|V|^2 \times |S_1| \times |T|)$, where $|T|$ denotes the cardinality of set T (*before* execution of the improvement method).

The second improvement method, which we call *label-and-delete,* is based on an idea by Dellaert (1984), who noticed that if some of the circuits found have edges in common, it may be possible to combine circuits by deleting some of those edges while retaining the property that every edge corresponding to a sensitive cell is contained in at least one of the circuits.

We have already shown that finding two completions D' and D'' of a table where $d'_{ij} \neq d''_{ij}$ for all (i, j) in S_1 is sufficient to show that the table is safe (recall Lemma 2.9 and Result 2.14). The improvement that we propose finds two completions of a table by making use of the following labeling procedure. Each edge (r_i, c_j) in the graph $G = (V, C)$ will be assigned a label L_{ij}. Initially, all labels have value zero. In the course of the heuristic, these labels will be updated whenever a circuit is found. Denote the circuit by

$$P = \{e_1, \ldots, e_q\},$$

where $e_1, \ldots, e_q$ represent the edges in P. In the labeling procedure, label-and-delete, all the edges in P are labeled in the following way. First, we choose (randomly) an edge in P. Because P is a circuit, we can without loss of generality assume that we have chosen e_1. We then add a value of 1 to the label of all edges e_j where j is odd (for $j = 1, \ldots, q$) and subtract 1 from the label of all edges e_j where j is even.

Now consider the table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S)$. The idea behind the label-and-delete algorithm is that, given any completion D' of this table for which $d'_{ij} = x_{ij}$ for all $(i, j) \notin S_1$, we can construct a new completion D'' by setting

$$d''_{ij} = d'_{ij} + L_{ij} \qquad \forall\, (i, j) \in S.$$

Now define

$$T = \{(i, j) \in S : d'_{ij} \neq d''_{ij}\} = \{(i, j) \in S : L_{ij} \neq 0\}.$$

Then, by Lemma 2.9, we know that the table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, T)$ is safe. Now if $S_1 \subset T$, then it is clear that $T \setminus S_1 \subset S_2$ is a set of complementary suppressions that makes the original table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe. In other words, if the labels corresponding to all sensitive cells (in S_1) are nonzero, then we can delete all edges with label zero from S_2. We can perform this improvement method either each time a circuit is found or only after all sensitive cells have been studied. It is even possible to incorporate it in the shortest-path algorithm by considering negative weight values for the cells that can be removed.

We end this section with an example of the label-and-delete algorithm. Consider the bipartite graph in Figure 4.

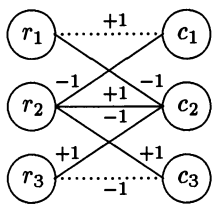

Figure 4. Example Label-and-Delete.

Here the dotted edges correspond to sensitive cells and the other edges correspond to complementary suppressions. The first circuit found by the heuristic was

$$P_1 = \{(r_1, c_1), (c_1, r_2), (r_2, c_2), (c_2, r_1)\},$$

and the labels added were $(+1, -1, +1, -1)$. The second circuit found was

$$P_2 = \{(r_3, c_3), (c_3, r_2), (r_2, c_2), (c_2, r_3)\},$$

and the labels added were $(-1, +1, -1, +1)$. Adding the labels corresponding to each of the circuits yields that the edges corresponding to the sensitive cells have a nonzero label, whereas (r_2, c_2) has label 0. We can now conclude that the complementary suppression corresponding to this edge is redundant.

5. RANDOMIZED ALGORITHM

In this section we construct a randomized algorithm, which consists of repeatedly running one of the heuristics. For the number or runs, two elements are important: a lower bound for the optimal solution and a stopping rule.

5.1 Sampling

In the heuristics for making a table safe described in the previous sections, we did not specify how we choose the next cell to be studied in each of the steps of the heuristics. Because it is not possible to give a suitable rule for this, we have chosen to study the cells in random order. This means that every time we perform the heuristic, we can get a different solution. Therefore, the heuristics can be seen to be so-called *randomized* algorithms. If we repeat the heuristic a number of times, then the best solution of this sample is usually better than just an arbitrary solution of the sample. In the next subsection we present several methods of determining a lower bound for the optimal solution. Of course, if the weight of the solution equals the lower bound, then we can stop sampling. We use this sampling idea in the following randomized algorithm to make a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe:

Randomized Algorithm.

Step 0. Compute a lower bound, say LB, for the weight of a set of complementary suppressions that makes the table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe.

Step 1. Use the heuristic to find a solution to the problem of making the table safe.

Step 2. If desired, use an improvement algorithm to improve the solution found in Step 1.

Step 3. STOP if the value of the solution obtained in the current iteration equals LB or if some stopping rule is satisfied. Otherwise, go to Step 1.

If the improvement step is used, then this algorithm can be compared to the multistart algorithm used for solving global optimization problems. That algorithm also proceeds by generating a set of locally optimal solutions and (after some stopping rule is satisfied) returns the best of those candidate solutions. For an extended description on the multistart concept, refer to the work of Rinnooy Kan and Timmer (1984).

5.2 Lower Bound

In this section we discuss two methods of determining a lower bound for the optimization problem of finding the set of complementary suppressions that makes a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe with minimum weight. Both methods are based on the observation that a sensitive cell $(i,j) \in S_1$ cannot be safe if it is the only sensitive cell in row i or in column j. Therefore, the following optimization problem yields a lower bound for our problem:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} s_{ij} \qquad \text{(P1)}$$

subject to

$$\sum_{i=1}^{m} s_{ij} \neq 1 \qquad \forall\, j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} s_{ij} \neq 1 \qquad \forall\, i = 1, \ldots, m,$$

$$s_{ij} = 1 \qquad \forall\, (i,j) \in S_1,$$

and

$$s_{ij} \in \{0, 1\} \qquad \forall\, (i,j) \in C \backslash S_1.$$

The set of cells H for which s_{ij} has value 1 in the optimal solution to this problem will correspond to the set of cells with minimum weight, containing the set of sensitive cells, and such that each row and each column of the table contains either no cell in H or at least two cells in H. Problem (P1) can be rewritten as a linear integer program by introducing two additional dummy vectors, $\mathbf{y}$ and $\mathbf{z}$:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} s_{ij} \qquad \text{(P2)}$$

subject to

$$\sum_{i=1}^{m} s_{ij} - m y_j \leq 0 \qquad \forall\, j = 1, \ldots, n,$$

$$\sum_{i=1}^{m} s_{ij} - m(y_j - 1) \geq 2 \qquad \forall\, j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} s_{ij} - n z_i \leq 0 \qquad \forall\, i = 1, \ldots, m,$$

$$\sum_{j=1}^{n} s_{ij} - n(z_i - 1) \geq 2 \qquad \forall\, i = 1, \ldots, m,$$

$$s_{ij} = 1 \qquad \forall\, (i,j) \in S_1,$$

$$y_j \in \{0, 1\} \qquad \forall\, j = 1, \ldots, n,$$

$$z_i \in \{0, 1\} \qquad \forall\, i = 1, \ldots, m,$$

and

$$s_{ij} \in \{0, 1\} \qquad \forall\, (i,j) \in C \backslash S_1.$$

In practice we will have to solve this problem by a branch-and-bound algorithm. In fact, we can use almost the same algorithm for the lower bound and the optimal solution. The only difference is that in the algorithm for the optimal, feasible solutions of Problem (P2) are only accepted if they yield a safe table. The foregoing formulation allows the use of standard software for solving integer programming problems. A linear programming relaxation of Problem (P2) does not look very promising, because it is such a typical 0-1 problem, especially with respect to table safety.

We know that in all rows with only one sensitive cell, at least one complementary suppression will be needed. From this property, a more rough lower bound can be deduced:

$$L_1 := \sum_{i:|\{j:(i,j)\in S_1\}|=1} \min_{j:(i,j)\in C\backslash S_1} \{w_{ij}\}.$$

A similar lower bound can be deduced if we consider all columns with only one sensitive cell:

$$L_2 := \sum_{j:|\{i:(i,j)\in S_1\}|=1} \min_{i:(i,j)\in C\backslash S_1} \{w_{ij}\}.$$

It is easy to see that $\max(L_1, L_2)$ is also a lower bound for the optimal value of (P). This is the lower bound that we will use in the randomized algorithm described earlier. In case of equal weights for all cells, this lower bound is exact, as was shown by Cox (1980).

5.3 Stopping Rule

In the foregoing algorithm we make use of a *Bayesian stopping rule.* Suppose that the set of values for the total weight of the complementary suppressions that can be obtained by performing a heuristic once (and, if desired, an improvement method) is given by $\{L, \ldots, U\}$, where the values of L and U are unknown. Assume further that each of the values in $\{L, \ldots, U\}$ has an unknown probability of being found by the heuristic. We denote the probability that the heuristic finds a solution with value $w \in \{L, \ldots, U\}$ by p_w. To make an approximative analysis possible, we then make the prior assumption that each value for L and U is equally likely and that for given values of L and U, each set of values for the probabilities $p_w\,(w = L, \ldots, U)$ is equally likely. Now given a sample of observations $w_1, \ldots, w_n$, we can compute certain posterior statistics. First, let

$$w^-(n) = \min_{i=1,\ldots,n} w_i$$

and

$$w^+(n) = \max_{i=1,\ldots,n} w_i.$$

One statistic that we can compute is the posterior probability that there exists a solution with a smaller value than $w^-(n)$. For $n \geq 2$, this probability is given by

$$\Pr(L < w^-(n)) = \frac{w^+(n) - w^-(n) + 1}{n + w^+(n) - w^-(n) - 1}.$$

Based on this probability, we can now propose the following stopping rule: Stop if the probability that a solution exists with a smaller value than the best solution found so far is smaller than some prescribed value p. For a more detailed description of this and other Bayesian stopping rules, refer to the work of Boender and Rinnooy Kan (1991).

6. NUMERICAL RESULTS

In the previous section we proposed a class of heuristics that make a table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1)$ safe. In this section we present the results obtained by the heuristics for some randomly generated sets of test problems. We compare the results with the optimal solution obtained by the branch-and-bound algorithm introduced in Section 4.1. In this algorithm we branch in each node by choosing an unsuppressed cell that we either suppress or forbid, until we have obtained a feasible solution for the lower bound Problem (P2) of Section 5.2. If this solution yields a safe table with lower weight than the best solution so far, then subtrees are pruned. We also present some of these intermediate results.

6.1 Random Generation of Tables

The tables that we used to test the algorithms that make a table safe were generated randomly. The weights for the nonsensitive cells were chosen uniformly in $\{1, \ldots, 10\}$. As we have seen, in our methods the values of the cells are not important for the obtained solution. These values were chosen uniformly in $\{-100, \ldots, 100\}$. The number of sensitive cells was also chosen randomly, depending on the dimension of the tables. In the first test we considered all versions of the heuristics previously described in a set of 10 test problems of dimension 20×20, where the number of sensitive cells was chosen uniformly from $\{1, \ldots, 40\}$. In the second test we considered only a selection of heuristics to observe their behavior in test problems of different dimensions and "density." In this test we considered two sets of tables, which we called sparse and dense. The number of sensitive cells in a table of dimension $d \times d$ was chosen uniformly in $\{1, \ldots, [\sqrt{d}]\}$ for sparse tables and in $\{d + 1, \ldots, 2d\}$ for dense tables, where $[x]$ denotes the value of x rounded to the nearest integer. The location of the sensitive cells was also generated uniformly in the table. We chose to test the selected algorithms on tables of dimension $d = 5$, 10, 20, and 40.

6.2 Results

Some of the steps in the proposed heuristics are done randomly. Therefore, it would be incorrect to base our conclusions on the results of just one run of the algorithm for a certain table. For all (randomly generated) tables, we performed a number of runs of the algorithm. We report the average results over this number of runs and over a set of tables of given dimension, for various values of the dimension. To make a good comparison possible, we considered the average weight ratio between heuristic solution and optimal solution. An indication of the accuracy of this ratio is given by the "error": the estimated standard deviation of the ratio divided by the square root of the total number of simulation runs. In the tables we report the results of the randomized algorithm using the stopping rule as described previously and the results of just one iteration of the algorithm. In this way, the profit of performing extra iterations can be compared to the extra amount of computer time. We use the delete-and-check improvement method only on the final solution obtained by the stopping rule and on the solution obtained by performing only one iteration of the randomized algorithm. Therefore, sometimes in the versions of the heuristic that perform this improvement method, the result after one iteration can be better than the results using the stopping rule. The value p used in the stopping rule was $.05n$. By this choice, the number of iterations n is always less than 21.

In the first test we performed 20 runs for each of the heuristics for a set of 10 test problems of dimension 20×20. This number of runs was large enough to make a first selection possible. In Tables 1–5 we denote each heuristic by a four-digit code of the form $\bar{w}_S/M/\mathrm{LD}/\mathrm{DC}$. Here $\bar{w}_S$ denotes the weight of the sensitive cells used for finding the circuits. We have tried $\bar{w}_S = 0$, $\bar{w}_S = -1$, and a random value (indicated by r) out of the set $-3, -2, -1, 0$. $M = 1$ if we use Dijkstra's algorithm to find the circuits, $M = 2$ if we use Dijkstra's algorithm in which the label-and-delete method has been incorporated, and $M = 0$ if we use the minimum rectangle method. LD refers to the label-and-delete improvement method: LD = 0 if this improvement method is not used, LD = 1 if it is used after all sensitive cells have been studied, and LD = 2 if it is used after each circuit found. Finally, DC = 0 if we do not use the delete-and-check improvement method, and DC = 1 if we do use it. The unit of time reported is 1/100 of a second (cs) on a 33-MHz 486 PC using Borland Pascal 7.0. The average value of the optimal solution in the first test turned out to be 12.6, and the average time to find it was about 200.000 cs. The solution of the lower bound problem (P2) yielded a safe table in 8 of 10 tables.

In Figure 5 the most efficient heuristics of this test and the intermediate results of the branch-and-bound procedure are put in a ratio–time diagram, in which we use a logarithmic scale for the time.

On the basis of the results from Table 1, we selected the most efficient heuristics, such that no other heuristic offered a better performance in the same amount of time. The following conclusions can be drawn from the first test:

- Using the minimum rectangle method for finding circuits gives very poor results, even if we use improvement methods and the stopping rule.
- The effects of the stopping rule on the results using Dijkstra's algorithm are quite large and depend on the $\bar{w}_S$ value: the average reduction of the deviation from the optimal solution is more than 60% for random $\bar{w}_S$, more than 40% for $\bar{w}_S = -1$, and about 30% for $\bar{w}_S = 0$.

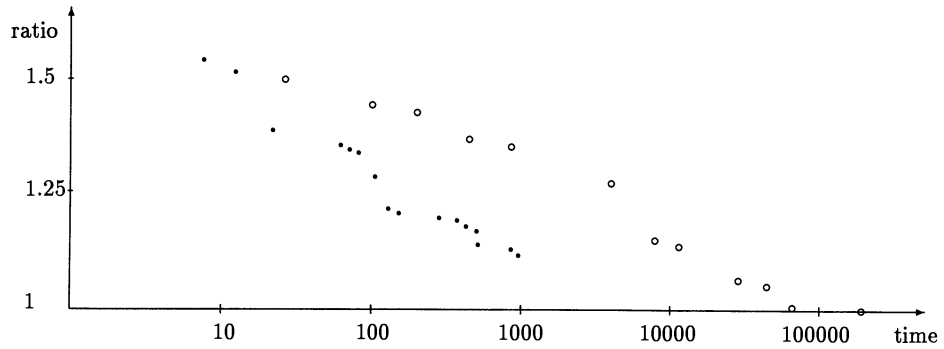

Figure 5. Performance of Selected Heuristics (Dots) and Branch-and-Bound Procedure (Circles) in the First Test.

- Using Dijkstra's algorithm, a negative value for $\bar{w}_S$ is worse if we perform only one iteration, but it is much better if we use the stopping rule.
- The delete-and-check improvement method is quite time-consuming but offers very good results; the average reduction of the deviation from the optimal solution is 35% for one iteration and 28% if we use the stopping rule.
- The effect of the label-and-delete improvement method is more or less the same when applied after each circuit or only at the end: an average reduction of the deviation of 31% if we use the stopping rule. If we perform one iteration, the reduction is (an additional!) 23% if we use delete-and-check and 17% if we do not use delete-and-check.
- Incorporating the label-and-delete improvement method in Dijkstra's method provides the largest reduction: about 35% for one iteration and about 40% if we use the stopping rule. It is also the most time-consuming heuristic.
- It took the branch-and-bound algorithm 12 times as long as the best heuristic to reach the same performance and 200 times as long to reach the optimum.

Table 1. Average Results of 20 Runs of the Heuristics Over 10 Tables, Where the Heuristics are Described by $\bar{w}_S$/M/LD/DC

	One iteration			Stopping rule			
Heuristic	Ratio	Error	Time	Ratio	Error	Time	Iter
r/1/0/0	1.727	.058	9	1.291	.025	104	11.4
r/1/0/1	1.494	.041	66	1.215	.021	150	11.4
r/1/1/0	1.638	.070	24	1.203	.016	285	11.4
r/1/1/1	1.384	.046	323	1.142	.012	517	11.4
r/1/2/0	1.585	.044	28	1.214	.020	341	11.4
r/1/2/1	1.357	.031	339	1.153	.014	580	11.4
r/2/2/0	1.436	.033	61	1.157	.015	703	10.8
r/2/2/1	1.294	.026	349	1.119	.011	945	10.8
−1/1/0/0	1.555	.055	8	1.346	.037	82	9.3
−1/1/0/1	1.353	.038	71	1.225	.026	129	9.3
−1/1/1/0	1.482	.037	24	1.251	.022	241	9.6
−1/1/1/1	1.269	.024	337	1.173	.018	505	9.6
−1/1/2/0	1.463	.034	28	1.228	.020	293	9.9
−1/1/2/1	1.264	.020	338	1.163	.015	566	9.9
−1/2/2/0	1.363	.026	61	1.173	.017	604	9.3
−1/2/2/1	1.224	.018	350	1.134	.014	852	9.3
0/1/0/0	1.527	.030	12	1.395	.024	108	8.5
0/1/0/1	1.364	.025	58	1.290	.020	146	8.5
0/1/1/0	1.397	.023	22	1.257	.016	196	8.6
0/1/1/1	1.281	.028	219	1.186	.013	373	8.6
0/1/2/0	1.434	.024	28	1.259	.016	244	8.5
0/1/2/1	1.292	.019	222	1.183	.012	424	8.5
0/2/2/0	1.390	.023	65	1.261	.016	569	8.5
0/2/2/1	1.265	.016	276	1.203	.015	759	8.5
0/0/0/0	2.372	.092	10	2.268	.089	78	7.7
0/0/0/1	2.016	.068	114	1.986	.067	173	7.7
0/0/1/0	2.242	.079	15	2.126	.072	143	8.4
0/0/1/1	1.792	.050	255	1.733	.046	356	8.4
0/0/2/0	2.276	.082	24	2.101	.069	213	8.7
0/0/2/1	1.782	.053	275	1.751	.047	429	8.7

NOTE: Ratio equals the total weight of suppressed cells using a heuristic divided by the total weight of suppressed cells for the optimal solution. Error denotes the standard deviation of the ratio divided by the square root of the number of simulation runs.

Table 2. Average Results of the Heuristics Over Five Tables, Where the Heuristics are Described by $\bar{w}_S/M/LD/DC/SR$

Dimension	5 × 5 sparse			5 × 5 dense		
Heuristic	Ratio	Error	Time	Ratio	Error	Time
0/0/0/0/0	1.156	.023	.2	1.458	.057	.3
−1/1/0/0/0	1.133	.023	.4	1.145	.024	.9
0/1/1/0/0	1.142	.023	.5	1.315	.034	1.3
−1/2/2/0/0	1.147	.023	.8	1.158	.024	2.7
r/1/0/0/1	1.030	.012	2.4	1.015	.006	3.8
−1/1/0/1/1	1.107	.021	3.2	1.016	.006	3.4
r/1/1/0/1	1.020	.010	3.3	1.013	.005	6.4
0/1/1/1/1	1.084	.018	3.6	1.060	.014	7.3
−1/1/1/1/1	1.081	.018	3.6	1.014	.004	6.7
r/1/1/1/1	1.016	.009	4.1	1.010	.005	7.0
−1/2/2/1/1	1.072	.015	5.3	1.019	.006	12.0
r/2/2/1/1	1.019	.009	6.3	1.006	.004	12.6
optimal	1		10.8	1		5.0

Table 4. Average Results of the Heuristics Over Five Tables, Where the Heuristics are Described by $\bar{w}_S/M/LD/DC/SR$

Dimension	20 × 20 sparse			20 × 20 dense		
Heuristic	Ratio	Error	Time	Ratio	Error	Time
0/0/0/0/0	1.350	.031	2	2.935	.069	11
−1/1/0/0/0	1.213	.026	3	1.700	.038	10
0/1/1/0/0	1.270	.029	4	1.263	.013	29
−1/2/2/0/0	1.195	.024	10	1.306	.019	72
r/1/0/0/1	1.124	.016	23	1.207	.009	128
−1/1/0/1/1	1.031	.007	27	1.353	.027	138
r/1/1/0/1	1.087	.011	29	1.153	.010	387
0/1/1/1/1	1.146	.017	31	1.125	.009	447
−1/1/1/1/1	1.043	.006	33	1.088	.010	682
r/1/1/1/1	1.054	.009	34	1.086	.008	713
−1/2/2/1/1	1.032	.006	80	1.072	.009	1,009
r/2/2/1/1	1.046	.006	81	1.065	.007	1,140
optimal	1		8,879	1		4,304

In Tables 2, 3, 4, and 5 we present the results of the selected heuristics. Each heuristic is denoted by a five-digit code of the form $\bar{w}_S/M/\mathrm{LD}/\mathrm{DC}/\mathrm{SR}$, where the meaning of the first four digits is identical to that in Table 1 and the fifth digit indicates whether we perform only one iteration (SR = 0) or the stopping rule (SR = 1). For each dimension and density, we considered 5 test problems and performed 20 runs per table, to obtain reasonably small values for the error.

On the basis of this second test, we can draw the following conclusions:

- The performance of the heuristics in the dense tables shows much more variation than in the sparse tables.
- In the sparse tables a random $\bar{w}_S$ value gives the best results. Because the required amount of computer time is always small in sparse tables, we can best use the $r/2/2/1/1$ heuristic, but the differences with other random $\bar{w}_S$ heuristics are small.
- In the dense tables it is necessary to use a label-and-delete heuristic in combination with the stopping rule, because the other heuristics perform much worse. Sometimes $\bar{w}_S = -1$ gives a better performance than a random value. Perhaps a random value out of the set $\{-2, -1, 0\}$ would give the best results. If the incorporated label-and-delete heuristic takes too much time, we can instead apply the heuristic only at the end.
- In the dense tables of smaller dimensions, the branch-and-bound algorithm is faster than the best heuristic. In other situations it is less attractive. Notice that the ratio for the best heuristic seems always acceptable.

7. SUMMARY AND CONCLUDING REMARKS

In this article we have considered the problem of avoiding disclosure in general two-dimensional tables. We have used the technique of cell suppression; that is, we try to find a set of complementary suppressions (having minimum weight, i.e., loss of information) that, if hidden, prevents the disclosure of the values of the sensitive cells. Because this problem is likely to be NP hard, we have concentrated on finding good heuristics. We have formulated a general heuristic that can be implemented in numerous ways. As the numerical results show, we can make choices that yield a *fast* heuristic, a *good* heuristic (in terms of weight value), or a combination of the two. Comparison of the solutions obtained by the heuristics with optimal values indicate that the best heuristics

Table 3. Average Results of the Heuristics Over Five Tables, Where the Heuristics are Described by $\bar{w}_S/M/LD/DC/SR$

Dimension	10 × 10 sparse			10 × 10 dense		
Heuristic	Ratio	Error	Time	Ratio	Error	Time
0/0/0/0/0	1.286	.025	1	2.897	.127	2
−1/1/0/0/0	1.250	.023	1	1.561	.041	3
0/1/1/0/0	1.237	.021	2	1.248	.024	7
−1/2/2/0/0	1.220	.020	3	1.272	.033	16
r/1/0/0/1	1.166	.018	8	1.093	.013	31
−1/1/0/1/1	1.203	.020	8	1.111	.012	26
r/1/1/0/1	1.140	.018	9	1.089	.012	79
0/1/1/1/1	1.182	.021	11	1.111	.014	85
−1/1/1/1/1	1.181	.021	10	1.045	.011	101
r/1/1/1/1	1.125	.017	11	1.067	.012	116
−1/2/2/1/1	1.183	.021	20	1.022	.006	163
r/2/2/1/1	1.128	.017	23	1.055	.010	199
optimal	1		752	1		106

Table 5. Average Results of the Heuristics Over Five Tables, Where the Heuristics are Described by $\bar{w}_S/M/LD/DC/SR$

Dimension	40 × 40 sparse			40 × 40 dense		
Heuristic	Ratio	Error	Time	Ratio*	Error	Time
0/0/0/0/0	1.412	.029	12	3.357	.097	84
−1/1/0/0/0	1.253	.014	17	2.054	.029	50
0/1/1/0/0	1.327	.021	22	1.356	.014	189
−1/2/2/0/0	1.297	.012	58	1.467	.024	492
r/1/0/0/1	1.073	.012	143	1.434	.019	919
−1/1/0/1/1	1.212	.008	116	1.476	.020	1,702
r/1/1/0/1	1.072	.009	171	1.267	.010	2,854
0/1/1/1/1	1.243	.014	161	1.175	.008	4,681
−1/1/1/1/1	1.171	.009	157	1.199	.011	7,894
r/1/1/1/1	1.061	.009	194	1.153	.008	7,165
−1/2/2/1/1	1.175	.009	412	1.133	.008	10,272
r/2/2/1/1	1.051	.008	496	1.142	.008	10,440
optimal	1		440,000	—	—	—

* Ratio compared to the best heuristic solution.

find a solution close to the optimal one. In dense tables of a small dimension, a branch-and-bound algorithm finding the optimal solution is preferable to the heuristics. In other situations heuristics will be preferred because of their speed. From the tests, it is also clear that the difference between good heuristics and the simple rectangle heuristic is enormous, implying that good heuristics can lead to a large decrease in loss of information. Essential elements of a good heuristic are the label-and-delete method, the delete-and-check method, and the application of the stopping rule in combination with the randomization.

Two extensions of the problem can be solved in an identical way. First, by considering the row and column totals and the grand total as new cells and by adding an extra row and column containing the new row or column totals, we can treat two-dimensional tables in which row/column suppressions are allowed in the same way. Second, we can treat three-dimensional tables in the same way, if the third dimension can take only two values (for instance, male–female or north–south). Suppressing a cell in one plane indicates automatically suppressing a cell in the other plane. Therefore, the three-dimensional table can be transformed in a two-dimensional table in which the weights of the cells are summed.

APPENDIX: PROOFS OF THE THEOREMS

Proof of Theorem 2.11

Consider some edge $(i, j) \in S_1$. By assumption, there exists a circuit in G containing (i, j). Denote this circuit by

$$P = \{e_1, \ldots, e_q\},$$

where $e_1 = (i, j)$. Because G is bipartite, we know that q is even. Given any completion, we can now increase the value of all cells e_i in P for which i is even by some value $\mathcal{H} > 0$ and decrease the value of all cells e_i in P for which i is odd by the same value. Because P is a circuit, we know that the modified values of the cells again yield a completion; that is, the modified value for $(i, j) \in S_1$ is consistent with the given row and column subtotals. Therefore, (i, j) is $\mathcal{H}$-safe. Because (i, j) was chosen arbitrarily from S_1, we can conclude that the table is $\mathcal{H}$-safe. The result now follows from the fact that $\mathcal{H}$ was chosen arbitrarily.

Proof of Theorem 2.12

Assume that $i \in A_N$. Then there is a $k \leq N$ with $i \in A_k$. According to the definition of B_k, it follows that $j \in B_k$ and thus also $j \in B_N$. Assume that $j \in B_N$. Then there is a $l \leq N - 1$ with $j \in B_l$. From the definition of A_l, it follows that $i \in A_{l+1}$ and thus i also belongs to A_N.

Proof of Theorem 2.13

Let D be a completion of table $(\mathbf{X}, \mathbf{a}, \mathbf{b}, S_1, S_2)$ and suppose that j_0 is not a marked-column for the element (i_0, j_0). Then

$$x_{i_0,j_0} = \sum_{i \in A_N} \sum_{j:(i,j) \in S} d_{ij} - \sum_{j \in B_N} \sum_{i:(i,j) \in S} d_{ij}$$

$$= d_{i_0,j_0} + \sum_{i \in A_N} \sum_{j \neq j_0:(i,j) \in S} d_{ij} - \sum_{j \in B_N} \sum_{i:(i,j) \in S} d_{ij} = d_{i_0,j_0}.$$

Because D has been chosen arbitrarily, it follows that (i_0, j_0) can be disclosed.

For the proof in the other direction, we assume that j_0 is a marked column. Let $L = \min\{k | j_0 \in B_k\}$; then there is an $i_L \in A_L$ with $(i_L, j_0) \in Y$ and a $j_L \in B_{L-1}$ with $(i_L, j_L) \in Y$. By means of induction, it follows that a circuit $\{(i_0, j_0), (i_0, j_1), (i_1, j_1), \ldots, (i_L, j_L), (I_L, j_0)\}$, consisting of elements of S, exists. According to Theorem 2.11, (i_0, j_0) cannot be disclosed.

[Received November 1992. Revised November 1993.]

REFERENCES

Boender, C. G. E., and Rinnooy Kan, A. H. G. (1991), "On When to Stop Sampling for the Maximum," *Journal of Global Optimization,* 1, 331–340.

Cox, L. H. (1976), "Statistical Disclosure in Publication Hierarchies," in *Proceedings of the American Statistical Association, Statistical Computing Section,* pp. 130–136.

——— (1977), "Suppression Methodology in Statistical Disclosure Analysis," in *Proceedings of the American Statistical Association, Social Statistics Section,* pp. 750–755.

——— (1978), "Automated Statistical Disclosure Control," in *Proceedings of the American Statistical Association, Survey Research Methods Section,* pp. 177–182.

——— (1980), "Suppression Methodology and Statistical Disclosure Control," *Journal of the American Statistical Association,* 75, 377–385.

Cox, L. H., Fagan, J., Greenberg, B., and Hemmig, R. (1986), "Research at the Census Bureau Into Disclosure Avoidance Techniques for Tabular Data," in *Proceedings of the American Statistical Association, Survey Research Methods Section,* pp. 388–393.

Dellaert, N. P. (1984), "Onderdrukking in Twee-Dimensionale Tabellen," internal note, Central Bureau of Statistics, Voorburg, The Netherlands.

DeSilets, L., Golden, B. L., Kumar, R., and Wang, R. (1991), "A Neural Network Model for Cell Suppression of Tabular Data," working paper, University of Maryland, College of Business and Management.

Duncan, G. T., and Lambert, D. (1986), "Disclosure-Limited Data Dissemination," *Journal of the American Statistical Association,* 81, 10–18.

Fellegi, I. P. (1972), "On the Question of Statistical Confidentiality," *Journal of the American Statistical Association,* 67, 7–18.

Garey, M. R., and Johnson, D. S. (1979), *Computers and Intractability, A Guide to the Theory of NP-Completeness,* New York: Freeman.

Gusfield, D. (1988), "A Graph Theoretic Approach to Statistical Data Security," *SIAM Journal on Computing,* 17, 552–571.

Harary, F. (1969), *Graph Theory,* Reading, MA: Addison-Wesley.

Kelly, J., Golden, B. L., and Assad, A. (1992), "Cell Suppression: Disclosure Protection for Sensitive Tabular Data," *NETWORKS,* 22, 397–417.

Rinnooy Kan, A. H. G., and Timmer, G. T. (1984), "Stochastic Methods for Global Optimization," *American Journal of Mathematical and Management Sciences,* 4, 7–40.