Information Sciences 238 (2013) 22–32

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A preprocessing optimization applied to the cell suppression problem in statistical disclosure control

Martin Serpell [a,*], Jim Smith [a], Alistair Clark [a], Andrea Staggemeier [b]

[a] University of the West of England, Frenchay Campus, Coldharbour Lane, Bristol BS16 1QY, United Kingdom
[b] Head of Center for Statistical and Analytical Intelligence, Office for National Statistics, Newport, United Kingdom

ARTICLE INFO

Article history:
Received 28 July 2010
Received in revised form 24 September 2012
Accepted 8 February 2013
Available online 13 March 2013

Keywords:
Information sharing
Statistical table
Cell suppression problem
Statistical disclosure control
Preprocessing optimization
Mathematical programming

ABSTRACT

As organizations start to publish the data that they collect, either internally or externally, in the form of statistical tables they need to consider the protection of the confidential information held in those tables. The algorithms used to protect the confidential information in these statistical tables are computationally expensive. However a simple preprocessing optimization applied prior to protection can save time, improve the resultant protection and on occasions enable the use of exact methods where otherwise heuristic methods would have been necessary. The theory behind this preprocessing optimization, how it can be applied and its effectiveness are described in this paper.

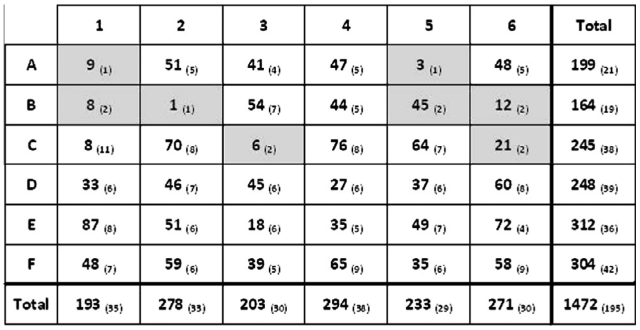
1. Introduction

How do we share information in the digital economy? Benefits can be obtained by pooling private data and then analyzing it, however this risks exposing private data. How to protect privacy in such circumstances is an active area of research [21,22]. This is a problem that has been addressed by National Statistics Agencies for some time but is now a problem that needs to be addressed by a wider variety of organizations. Much information that is made public is done so in the form of statistical tables as publishing the source data would break confidentiality. Each cell in the statistical table will typically contain an aggregate of some information, for example it could be the total cost of a given drug dispensed in a given area of a country. The statistical table will also have row and column totals that are referred to as marginals. Like National Statistics Agencies, many organizations are obliged to maintain the confidentiality of the information they hold. If confidentiality is compromised then organizations can lose the co-operation and goodwill of their data contributors and may even be subjected to legal action.

Confidentiality can be compromised if, for example, there is a cell in the published statistical table that is not an aggregate of data from many contributors but from just one or two. To illustrate this point in Figs. 1–3 the cell data is represented in the format $X_{(y)}$ where X is the sum of the data provided by y contributors. In Figs. 1–3 any cell with less than or equal to two contributors has been suppressed to ensure the confidentiality of those contributors. If the information in the cell can be traced back to source then confidentiality has been compromised. Typically in this situation the cell in question is left blank in the published statistical table and this cell is called a primary suppressed cell.

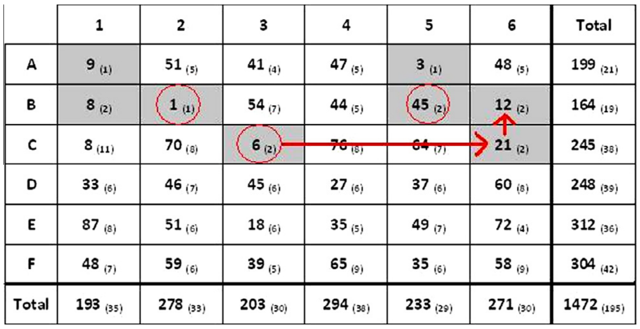
* Corresponding author.
E-mail address: Martin2.Serpell@uwe.ac.uk (M. Serpell).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.02.006



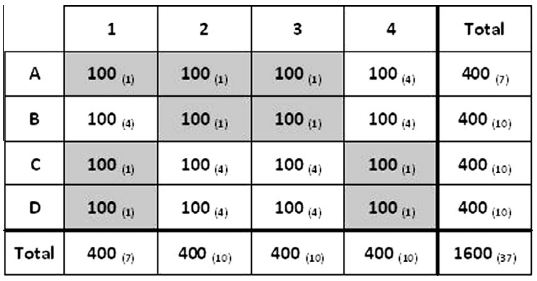
	1	2	3	4	5	6	Total
A	9 (1)	51 (5)	41 (4)	47 (5)	3 (1)	48 (5)	199 (21)
B	8 (2)	1 (1)	54 (7)	44 (5)	45 (2)	12 (2)	164 (19)
C	8 (11)	70 (8)	6 (2)	76 (8)	64 (7)	21 (2)	245 (38)
D	33 (6)	46 (7)	45 (6)	27 (6)	37 (6)	60 (8)	248 (39)
E	87 (8)	51 (6)	18 (6)	35 (5)	49 (7)	72 (4)	312 (36)
F	48 (7)	59 (6)	39 (5)	65 (9)	35 (6)	58 (9)	304 (42)
Total	193 (35)	278 (33)	203 (30)	294 (38)	233 (29)	271 (30)	1472 (195)

Fig. 1. An example 6 × 6 statistical table. The eight primary cells have been highlighted with shading. The number of contributors to each cell is given in brackets.



	1	2	3	4	5	6	Total
A	9 (1)	51 (5)	41 (4)	47 (5)	3 (1)	48 (5)	199 (21)
B	8 (2)	1 (1)	54 (7)	44 (5)	45 (2)	12 (2)	164 (19)
C	8 (11)	70 (8)	6 (2)	76 (8)	64 (7)	21 (2)	245 (38)
D	33 (6)	46 (7)	45 (6)	27 (6)	37 (6)	60 (8)	248 (39)
E	87 (8)	51 (6)	18 (6)	35 (5)	49 (7)	72 (4)	312 (36)
F	48 (7)	59 (6)	39 (5)	65 (9)	35 (6)	58 (9)	304 (42)
Total	193 (35)	278 (33)	203 (30)	294 (38)	233 (29)	271 (30)	1472 (195)

Fig. 2. An example 6 × 6 statistical table with the sequence in which suppressed cells are exposed using an unpicking algorithm. The eight primary cells have been highlighted with shading. The number of contributors to each cell is given in brackets.



	1	2	3	4	Total
A	100 (1)	100 (1)	100 (1)	100 (4)	400 (7)
B	100 (4)	100 (1)	100 (1)	100 (4)	400 (10)
C	100 (1)	100 (4)	100 (4)	100 (1)	400 (10)
D	100 (1)	100 (4)	100 (4)	100 (1)	400 (10)
Total	400 (7)	400 (10)	400 (10)	400 (10)	1600 (37)

Fig. 3. An example 4 × 4 statistical table, provided by Lawrence Cox, that cannot be unpicked. The nine primary cells have been highlighted with shading. The number of contributors to each cell is given in brackets.

It is the duty of the National Statistics Agency to determine which cells should be protected. For each primary suppressed cell the National Statistics Agency will also determine a range within which the value of the primary suppressed cells must not be able to be calculated. This range is bounded by the lower protection level, *lpl*, and the upper protection level, *upl*. The values assigned to the *lpl* and *upl* are not published. This alone however does not guarantee confidentiality as the cell value may possibly be calculated by subtracting the other cell values in the same row/column from the row/column total. Therefore to guarantee the confidentiality of the information being published in a statistical table it is necessary to suppress other cells in the statistical table. These other suppressed cells are known as secondary suppressed cells. The combination of primary and secondary suppressed cells are known as a suppression pattern. Finding the lowest cost suppression pattern is known as the cell suppression problem in statistical disclosure control, see [23,14] for more details.

The cell suppression problem is a member of the class of NP-hard problems when solving for optimality. In fact, the problem of finding a secondary suppression pattern is easy to be achieved, for example if all cells are suppressed this is a feasible pattern but clearly not optimal. It is when solving the cell suppression problem optimally that as the size of the table to be protected grows the number of possible solutions that need to be evaluated grows much quicker. For a table with n cells there are $2^n - 1$ possible suppression patterns. Because of the very large number of constraints that define the cell suppression problem ILP techniques can only find the optimal solution for small and medium sized statistical tables. The protection of 2-dimensional statistical tables has been well catered for. Fischetti and Salazar-Gonzalez [11] and Castro [2] developed a integer programming (IP) network flow model that can optimally protect 2-dimensional non-hierarchical tables with up to 500,000 cells and near optimally protect 2-dimensional hierarchical tables with up to 250,000 cells. Protecting statistical tables with more than two dimensions has been more problematic. An IP model initially proposed by Kelly et al. [16] and reformulated by Fischetti and Salazar-Gonzalez [11] has been used to optimally protect 2^+-dimensional hierarchical tables with 10,000 cells however this technique can be unreliable due to the limitations of the mathematical solver. A modular approach [7] that partitions tables prior to their protection more reliably protects 2^+-dimensional hierarchical tables with 50,000 cells but the price for doing this is slight over-suppression. A heuristic linear programming (LP) model that protects cells one at a time was developed by Kelly et al. [16] and has been shown to reliably protect 2^+-dimensional hierarchical tables with up to 40,000 cells. This model however will not scale and over-suppresses. Non-mathematical programming techniques such as Hypercube [15] have been developed to quickly protect 2^+-dimensional hierarchical tables with up to 500,000 cells however these too are prone to over-suppression. Approaches like Modular and Hypercube that partition the tables before adding protection have to backtrack through the reconstituted table to account for linkages and then add extra protection, this approach has already been reported by Cox [5,6] not to guarantee full protection.

This paper takes a complementary approach – namely developing techniques that can be used to reduce the size of the problem, regardless of the formulation or solution approach. It is known that removing anything that is redundant from the mathematical program can make an efficiency gain. For example redundant equations, variables and protection levels can be removed. Another preprocessing efficiency gain can be obtained by removing any table cells that have the value set to zero or whose values must be published, subject to adjusting any marginal totals necessary. This decreases the number of working variables and constraints that the solver requires to find a solution, which in turn allows larger statistical tables to be protected.

Linear programming models and local search algorithms are used on relaxed cell suppression problems to obtain near optimal solutions when integer programming models are infeasible. Some have moved away from trying to calculate the optimal solution and have instead employed heuristic techniques to find near optimal solutions quickly. Others have employed hybrid algorithms that combine linear programming and heuristic techniques [3,4].

Anecdotally this has been used, but we have been unable to find documentation in the literature. In a previous paper [19] we reported some preliminary findings using an initial formulation. This paper builds on that work as follows. Research subsequent to [19] revealed that the initial formulation was proof against "unpicking" by the modified version of the shuttle algorithm [1] but in certain circumstances could be shown to vulnerable to approaches that attack multiple cells in tandem. Here we provide a corrected formulation with proofs of correctness. However, this necessitates a more complex approach to the rapid identification of the set K. We therefore compare several alternatives, looking at both experimental evidence and run-time complexity analysis to motivate a choice of suggested method. We then fully evaluate the effectiveness of our proposed technique to examine the reduction in problem size and hence the performance gains in terms of reduced run-times and reduced information loss in the protected tables. Building on our findings subsequent to the publication of [19] in this paper we now consider three different Cell suppression algorithms, and crucially we examine far more complex tables with hierarchical dimensions.

Section 2 presents the theory behind this preprocessing optimization. Section 3 describes how to find candidates for initially exposed primary suppressed cells. Section 4 evaluates the use of this preprocessing optimization with different cell suppression algorithms. Section 5 contains our conclusions and Section 6 lists further research.

2. An introduction to the theory behind this preprocessing optimization

2.1. Terminology

A statistical table with marginal totals can be represented as a set of cells, please see details of the model in [3,10,12], $a_i,\ i = 1,\dots,n$, satisfying m linear constraint equations such that $Ma = 0$, where M_{ij} has one of the values $\{0, +1, -1\}$.

$$\sum_{i=1}^{n} M_{ij} a_i = 0, \quad j = 1,\dots,m \tag{1}$$

These linear constraint equations represent the relationships between the cells in the statistical table and the marginal (row and column) totals. The statistical agency will define a set P of primary cells whose publication will be suppressed in order to protect the confidentiality of the contributors to those cells. The statistical agency will provide lower and upper protection levels (lpl and upl) for each cell in P such that an external attacker must not be able to calculate a_p within the range lpl_p to upl_p. In our example statistical tables in Figs. 1–3 the lower protection level for each primary cell is its value minus 10% and the upper protection level for each primary cell is its value plus 10%. For a_p to be safe

$$\underline{a_p} \leqslant lpl_p \quad \text{and} \quad \overline{a_p} \geqslant upl_p \tag{2}$$

where $\underline{a_p}$ is the lower bound and $\overline{a_p}$ the upper bound of the feasible range that the external attacker can calculate for a_p if only the primary cells P have been suppressed [3].

$$\begin{aligned} \underline{a_p} = \min \quad & x_p & \qquad \overline{a_p} = \max \quad & x_p \\ s.t. \quad & Mx = 0 \quad \text{and} & s.t. \quad & Mx = 0 \\ & x_i \geqslant 0, \quad i \in P & & x_i \geqslant 0, \quad i \in P \\ & x_i = a_i, \quad i \notin P & & x_i = a_i, \quad i \notin P \end{aligned} \tag{3}$$

If the external attacker is able to calculate $\underline{a_p} > lpl_p$ or $\overline{a_p} < upl_p$ then a_p is unsafe (a_p can be disclosed). In our example statistical tables in Figs. 1–3 a primary cell is unsafe if its feasible range can be calculated to within 10% of its actual value. It should be noted that we are considering the external attacker on tables which have not yet been protected by secondary suppressed cells in order to gauge the level of disclosiveness of the tables for our preprocessing optimization. The time complexity for doing this calculation is $O(cp^2)$ where c is the number of constraints that describe the relationships between the cells in the table and p is the number of primary cells [4,20].

Noting that some primary cells may occur alone in a marginal total, whereas others (e.g. those sharing rows/columns) may effectively protect each other, we define the following partition of the set of primary cells P.

An *exposed* primary suppressed cell in an unprotected statistical table with marginal totals is one whose value can be calculated, within a given lower and upper protection limit, by an external attacker when only the primary cells P have been suppressed. That is to say, p is a member of the set E of *exposed* primary suppressed cells if $\underline{a_p} > lpl_p$ or $\overline{a_p} < upl_p$. $E \subseteq P$.

A *not exposed* primary suppressed cell in an unprotected statistical table with marginal totals is one whose value cannot be calculated, within a given lower and upper protection limit, by an external attacker when only the primary cells P have been suppressed. That is to say, p is a member of the set N of *not exposed* primary suppressed cells if $\underline{a_p} \leqslant lpl_p$ and $\overline{a_p} \geqslant upl_p$. $N \subseteq P$, $E \cup N = P$ and $E \cap N = \emptyset$. The reason why there may be *not exposed* primary suppressed cells in an unprotected statistical table is due to their locations in that table. Each *not exposed* primary suppressed cell receives sufficient protection from other primary suppressed cells in the table to prevent an external attacker from being able to calculate a feasible range of values within the given protection level.

This property was observed by Fischetti and Salazar-González [10] and used as the basis of a preprocessing phase to reduce the size of the Cell Suppression Problem. We now go onto extend this preprocessing optimization further.

Let E_u be the set of primary suppressed cells in an unprotected statistical table whose values can be compromised by "unpicking" at the primary suppressed cells [1,8,9]. Where "unpicking" is the iterative process of deducing suppressed cell values using the associated constraint equations. Members of E_u can be exposed using a single constraint equation. For example, the value of the cell $B2$ in the statistical table in Fig. 1 can easily be calculated exactly using the other cell values in column 2. In a similar way the value of the cell C3 can be calculated exactly using the other cell values in column 3. Once the value of C3 has been calculated then the value of C6 can be calculated exactly using the other cell values in row C. Once C6 has been calculated then the value of B6 can be calculated exactly using the other cell values in column 6. The value of cell B5 can be compromised as its maximum value can be calculated as 48 by assuming that the value of A5 is zero and then subtracting the non-suppressed values from the column total in column 5. As this is less than its upper protection level of 49.5 (its actual value plus 10%) cell B5 is *partially exposed*.

In the case of the unprotected statistical table in Fig. 1 the complete set of primary suppressed cells whose values can be compromised, E, and the set of primary suppressed cells whose values can be compromised by unpicking at the suppressed cells, E_u, are the same, i.e. $E = E_u$. In the set E are primary suppressed cells whose values can be exactly calculated and we have named these cells *fully exposed* primary suppressed cells. The primary suppressed cells in E whose values cannot be exactly calculated we have named *partially exposed* primary suppressed cells. In the statistical table in Fig. 1 the cells $\{B2, B5, B6, C3, C6\}$ are *exposed* suppressed cells and the cells $\{A1, A5, B1\}$ are *not exposed* primary suppressed cells. Further the cells $\{B2, B6, C3, C6\}$ are *fully exposed* primary suppressed cells and the cell $\{B5\}$ is a *partially exposed* primary suppressed cell. Cell $B5$ being *partially exposed* as its value cannot be calculated exactly but it can be calculated within the given upper protection level. There has been discussion in the statistical disclosure control community as to whether or not the feasible range of each primary suppressed cell should be published, for further information see [13,6].

As can be seen in Fig. 2 the values of some of the *exposed* primary suppressed cells in E can have their values calculated immediately whereas others must wait until some other member of E has had its value calculated. In our example the set of primary suppressed cells $\{B2, B5, C3\}$ can have their values calculated, either exactly or within the given protection level, immediately as described above. Therefore they have been named *initially exposed* primary suppressed cells, this set is referred to as I. The set of primary suppressed cells $\{B6, C6\}$ can only have their vales calculated after some other member of E has had its value calculated and have been named *consequentially exposed* primary suppressed cells, this set is referred to as C. $I \cup C = E$ and $I \cap C = \emptyset$. If an external attacker exposes a primary suppressed cell then it is for one of two reasons, the cell is either *initially* or *consequentially* exposed. If $I = \emptyset$ then both $C = \emptyset$ and $E = \emptyset$.

In the case of the unprotected statistical table that is shown in Figs. 1 and 2 $E = E_u$, however this is not always the case. Lawrence Cox of the National Center for Health Statistics in the USA [6] provided an example of an unprotected statistical table, shown in Fig. 3, where $E \neq E_u$. In this unprotected statistical table cell $A1$ is an *exposed* primary suppressed cell, a member of E, and as it is the only *exposed* primary suppressed cell it is an *initially exposed* primary suppressed cell and hence a member of set I. Yet the value of $A1$ cannot unpicked from the unprotected statistical table. The value of the nine primary suppressed cells can be calculated to be 900 by subtracting the value of the published cells $\{A4, B1, B4, C2, C3, D2, D3\}$ from the grand total of 1600. $A1$ is *fully exposed* because $A1 = 900 - A2 - B2 - A3 - B3 - C1 - C4 - D1 - D4$ which can be rearranged as $A1 = 900 - (A2 + B2) - (A3 + B3) - (C1 + C4) - (D1 + D4)$ and as the value of each pair of cells in brackets can be calculated using their column and row totals therefore $A1$ can be calculated exactly to have the value 100.

Fig. 3 is an example of an unprotected table which shows that $\exists x \in I \cdot x \notin E_u$. Let E_n be the set of *exposed* primary suppressed cells that cannot be found by unpicking an unprotected statistical table, then $E_n \cup E_u = E$ and $E_n \cap E_u = \emptyset$. Members of E_n can only be exposed by simultaneously using multiple constraint equations. Let I_u be the set of *initially exposed* primary suppressed cells that can be found by unpicking an unprotected statistical table and let I_n be the set of *initially exposed* primary suppressed cells that cannot be found by unpicking an unprotected statistical table, then $I_n \cup I_u = I$ and $I_n \cap I_u = \emptyset$. Also $I_u \subseteq E_u$ and $I_n \subseteq E_n$. Therefore $\forall x \in I \cdot x \in E_n \cup E_u$.

In the unprotected statistical table that is shown in Fig. 1 and 2 $E_u = \{B2, B5, B6, C3, C6\}, E_n = \emptyset$, $I_u = \{B2, B5, C3\}$ and $I_n = \emptyset$ and in the unprotected statistical table that is shown in Fig. 3 $E_u = \emptyset$, $E_n = \{A1\}$, $I_u = \emptyset$ and $I_n = \{A1\}$.

2.2. Finding an upper limit for the number of initially exposed cells

Let I_{max} be the maximum number of initially exposed cells that a statistical table can contain. Then for a 1-dimensional non-hierarchical statistical table $I_{max} = 1$. If there were more than one primary suppressed cell in a 1-dimensional non-hierarchical statistical table it would be impossible to calculate either cell value exactly. It may, however, be possible to calculate the feasible lower or upper bounds of one of the cells to be within the given protection levels. For a $n * m$ 2-dimensional non-hierarchical statistical table $I_{max} = (n - 1) + (m - 1)$ as this is the maximum number of primary suppressed cells that can occupy a row or column without there being another primary suppressed cell in it. For a 3-dimensional non-hierarchical statistical table it is easiest to imagine these as occupying the faces but not the edges of a hyper-rectangle except for a single face where the four corners are also primary suppressed, of course there are actually many permutations of the positions. So, for a $n * m * l$ 3-dimensional non-hierarchical statistical table $I_{max} = (n - 1)(m - 1) + (m - 1)(l - 1) + (n - 1)(l - 1) + 4$.

If T is the number of cells in a $n * m$ 2-dimensional non-hierarchical statistical table then $I_{max} = (n - 1) + ((T/n) - 1)$. Taking the derivative with respect to n we have $\frac{d}{dn} I_{max} = 1 - \frac{T}{n^2}$, which is zero when $n = \sqrt{T}$ indicating that there is a turning point for I_{max} when the 2-dimensional non-hierarchical statistical table is square. Taking the second derivative with respect to n we have $\frac{d^2}{dn^2} I_{max} = \frac{2T}{n^3}$ which is positive so the gradient is always increasing indicating that the turning point is a minima. For example, take a 2-dimensional non-hierarchical statistical table with 100 cells. For a $50 * 2$ table $I_{max} = 50$ and for a $10 * 10$ table $I_{max} = 18$, the square table having the lowest maximum number of initially exposed cells.

For higher dimension non-hierarchical statistical tables we can reason that I_{max} is related to the surface area of the statistical table. We know that in any dimension the shape that has the lowest surface area for a given volume is a hypersphere which for hyper-rectangular statistical tables is best approximated by a hypercube. Therefore I_{max} is minimized for square or more generally hyper-cubic statistical tables.

The number of constraint equations, $|J|$, that describe a $n * m$ 2-dimensional non-hierarchical statistical table is $n + m$ and as $I_{max} = (n - 1) + (m - 1)$ we can see that $I_{max} \leqslant |J|$. This is true for any dimension statistical table including hierarchical statistical tables.

2.3. Candidates for initially exposed cells

If I is the set of *initially exposed* cells then let K be the set of primary suppressed cells that are *candidates* for being in the set I such that $I \subseteq K$. The size of set of *initially exposed* primary suppressed cells is often much smaller than the size of set of primary suppressed cells. As $|I| \leqslant |K| \leqslant |P|$ is always true and $|I| \ll |P|$ is often true we want to find K such that $|I| \leqslant |K| \ll |P|$. K can be considered as a rough set bounded by I and P [17,18]. Unusually in this case I is unknown and the task is to use K to get as close to I as possible by finding K with minimum $|K|$.

Fig. 3 has demonstrated that unpicking is not the only way to find the values of primary suppressed cells. Let I_u be the set of *initially exposed* primary suppressed cells that can be found using an unpicking algorithm and K_u be the set of *candidate* cells for I_u that can be found using an unpicking algorithm such that $I_u \subseteq K_u$. Fig. 3 showed that $\exists x \in I \cdot x \notin K_u$. Therefore in order to ensure that all I have been found then the set of *initially exposed* primary suppressed cells that cannot be found using an unpicking algorithm, I_n, must be found. Let us define a set K_n to be the set of cells that are *candidate*s for I_n such that $I_n \subseteq K_n$. Then $I = I_u \cup I_n$, $K = K_u \cup K_n$ and $I \subseteq K$.

In order to find K, K_u can be found using an unpicking algorithm, K_n must be found. E is the set of *exposed* primary suppressed cells, all of these cells can be found using an external attacker Linear Program. However if an unpicking algorithm is used to carry out an external attack on an unprotected statistical table then a subset, E_u, of E is found such that $E_u \subseteq E$. As the set I_n must be contained in the set $E \setminus E_u$, K_n can be defined as the set $E \setminus E_u$ which means we can now find K which in turn contains I.

Finding both E, by a Linear Program, and E_u, by an unpicking program, can be done in polynomial time therefore finding K_n can be done in polynomial time. As finding both K_u and K_n can both be done in polynomial time finding K can be done in polynomial time.

2.4. Proposition

Only the *initially exposed* primary suppressed cells, I, need to be protected by suppressing secondary cells in order to make a published statistical table safe from an external attacker.

This is true as the *not exposed* primary suppressed cells, N, are already safe from an external attacker and once the *initially exposed* primary suppressed cells have been protected the *consequentially exposed* primary suppressed cells, C, are also safe. See the definitions of $I,\ C$ and N in Section 2.1.

It is worth noting that even if it makes great improvements in some cases, this preprocessing stage does not change the NP-hard nature of the cell suppression problem, since in the worst case $I = P$, and we also now have to solve the problem of finding I.

A Corollary to this proposition is that if $I = \emptyset$ then $N = P$ and therefore the statistical table is already adequately protected.

3. Finding candidate initially exposed primary cells

The objective of this section is to compare the performance of different algorithms that identify candidates for initially exposed primary cells.

3.1. The algorithms

The purpose of these algorithms is to find the smallest subset of cells, K, in a statistical table with margin totals that contain all of the *initially exposed* primary suppressed cells, such that $I \subseteq K \subseteq E \subseteq P$.

The Algorithms that are considered here for finding candidates for *initially exposed* primary suppressed cells are given in Fig. 4.

Where

E = Exposed primary cells.
Ef = Fully exposed primary cells.
Ep = Partially exposed primary cells.
Eu = Primary cells that can be exposed using an unpicking algorithm.
$K_n = E_n = E \setminus Eu$.

K_1 = Primary cells that are found singly in at least one constraint equation.

K_{pl} = Primary cells that share their constraint equations with other primary cells but whose protection range is greater than the sum of the protection ranges of the other primaries in the constraint equations.

K_{nv} = Primary cells that share their constraint equations with other primary cells but whose protection range is greater than the sum of the nominal values of the other primaries in the constraint equations.

K_s = Primary cells that share their constraint equations with other primary cells but whose protection range and/or nominal value is greater than three times the sum of the nominal values of the other primaries in the constraint equations.

Finding $E,\ Ef,\ Ep$ and K_n all require the use of a mathematical program to solve the external attacker model, therefore each has a time complexity of $O(cp^2)$. Eu however only requires the unpicking algorithm which has a time complexity of $O(cp)$. Similarly K_s also has a time complexity of $O(cp)$. K_1 is the simplest set of primary cells to find and correspondingly has a time complexity of $O(c)$.

Algorithm $K1$ simply collects a set of primary suppressed cells that might be at risk of being exposed. $K3$ augments K_1 with the primary cells that can be partially exposed using a linear program and with the primary cells that can be exposed using a linear program but not an unpicker. $K5$ augments K_1 with the primary cells that cannot be protected by merely suppressing the other primaries in their row or column (K_{pl}, K_{nv}) and with the primary cells that can be exposed using a linear program but not an unpicker.

3.1.1. Finding K_{pl}

The set K_{pl} can be found without the need to use a Mathematical Programming Solver or an Unpicking Program, it has a time complexity of $O(cp)$.

$$\begin{aligned}
&Algorithm\ E: && K = E\\
&Algorithm\ K1: && K = K_1 \cup (K_s \cap E)\\
&Algorithm\ K3: && K = K_1 \cup Ep \cup K_n\\
&Algorithm\ K5: && K = K_1 \cup (K_{pl} \cap E) \cup (K_{nv} \cap E) \cup K_n
\end{aligned}$$

Fig. 4. Algorithms for finding candidates for initially exposed primary cells.

Let Q be the set of cells that are not in P, these cells do not require protection. Let

$$c_j = \sum_{i \in Q} M_{ij} a_i, \quad j = 1, \ldots, m \tag{4}$$

then,

$$c_j + \sum_{i \in P} M_{ij} a_i = 0, \quad j = 1, \ldots, m \tag{5}$$

A necessary condition for us to establish that $p \in K$ is the existence of at least one constraint equation in which the amount of "uncertainty" (and hence protection) provided by the given lower and upper protection levels of the other suppressed primary cells in that constraint equation is less than the required protection limits for p. For each element $p \in P$ let J denote the set of linear constraint equations (equivalent to rows of M) in which p participates, i.e. $\forall j \in J \cdot M_{pj} \neq 0$. For each $j \in J$ let H_j be the set of primary cells in j. For each $p \in P$,

$$\underline{a'_p} = max_{j \in J} \left(-c_j - \sum_{i \in H_j \setminus p} M_{ij} upl_i \right) \tag{6}$$

$$\overline{a'_p} = min_{j \in J} \left(-c_j - \sum_{i \in H_j \setminus p} M_{ij} lpl_i \right) \tag{7}$$

Then we can say that p is a candidate member of the set I of *initially exposed* primary suppressed cells if $\underline{a'_p} > lpl_p$ or $\overline{a'_p} < upl_p$. The set, K, of candidate members of I contains the set I. This is because the values of $\underline{a_p}$ and $\overline{a_p}$ that the external attacker can calculate cannot be better than $\underline{a'_p}$ and $\overline{a'_p}$, but are likely to be worse.

$$\underline{a_p} \leqslant \underline{a'_p} \quad \text{and} \quad \overline{a_p} \geqslant \overline{a'_p} \tag{8}$$

3.1.1.1. Example. Taking a 6 by 6 unprotected statistical table with marginal totals (Fig. 2) as an example, the process of finding K can be shown. In our example the statistical agency has defined $P = \{A1, A5, B1, B2, B5, B6, C3, C6\}$. When the test for the *fully exposed* primary suppressed cells is applied five primary cells are found to be exposed, $E = \{B2, B5, B6, C3, C6\}$ and therefore $N = \{A1, A5, B1\}$. The values of cells $B2$, $B6$, $C3$ and $C6$ are calculated exactly and the feasibility range of cell B5 is calculated within its lower and upper protection levels which in this case is 10% of the cell's value.

By contrast applying the test for *candidates* for *initially exposed* primary suppressed cells, K, (see Table 1) we find that $K = \{B2, B5, C3, C6\}$.

For this preprocessing optimization to work it is not necessary (nor is it possible) to determine which cell is in C and which is in N.

3.1.2. Finding K_{nv}

For each element $p \in P$ let J denote the set of linear constraint equations (equivalent to rows of M) in which p participates, i.e. $\forall j \in J \cdot M_{pj} \neq 0$. For each $j \in J$ let H_j be the set of primary cells in j. Then we can say that p is a candidate member of the set I of *initially exposed* primary suppressed cells if for each $p \in P$,

$$upl_p > \sum_{i \in H_j \setminus p} M_{ij} a_i \tag{9}$$

Finding K_{nv} has a time complexity of $O(cp)$.

Table 1
Workings to find members of the K. Any primary cell that is at risk of having its protection range violated is a member of K.

Cell	Protection range	$\underline{a'_p}$	$\overline{a'_p}$	Protected
8	8–10	8	10	Yes
12	2–4	2	4	Yes
15	7–9	7	9	Yes
16	0–2	1	1	No
19	40.5–49.5	44	46	No
20	10.8–13.2	9.9	14.1	Yes
24	5–7	6	6	No
27	18.9–23.1	19.8	22.2	No

Table 2
Distribution of experimental statistical tables, used in exploring subsets of primary cells, by number of dimensions and number of dimensions that are hierarchical.

Dimensions	Hierarchical dimensions					
	0	1	2	3	4	Total
2	155	137	1	0	0	293
3	20	0	0	0	0	20
4	0	0	0	0	0	0
Total	175	137	1	0	0	313

3.2. An experiment to find out which algorithm generates the smallest set of candidate initially exposed cells

There are difficulties in obtaining a large enough quantity of real statistical tables from which statistical inferences can be made. The most obvious problem being that these tables contain confidential information and that access to unprotected tables is therefore restricted by National Statistics Agencies. To get around this problem bootstrapping was used to generate 200 three-dimensional non-hierarchical tables from a single real world 3-dimensional non-hierarchical table. The tables ranged in size from 6100 to 26,775 cells and had between 11.7% and 15.2% primary cells. The candidate initially exposed primary cells were identified using algorithms E and $K5$. The average reduction in the number of primary cells requiring protection was 14.95% ($\sigma = 4.31\%$) for algorithm E and 22.98% ($\sigma = 4.87\%$) for algorithm K5. Using bootstrapping however did not allow the algorithms to be tested against hierarchical tables or for the factors describing the tables to be greatly varied. This restricted the usefulness of this approach.

In order to examine the performance of each algorithm on a wider variety of tables with different properties synthetic statistical tables were generated. This allowed a fractional factorial experimental design to be chosen to find which algorithm generates the smallest set of candidate initially exposed cells. A set of 313 statistical tables were used with the factor values randomly assigned. Table 2 shows how the statistical tables vary with number of dimensions and number of dimensions that are hierarchical.

Each of these statistical tables was protected using a SAS/OR implementation of the Kelly et al. [16] ILP using the candidates for initially exposed primary cells produced by each of the algorithms under investigation. The SAS version used was SAS 9 solver with SAS/OR Opt module. There are a variety of solvers in SAS and OptMILP was used. Two response variables were measured for each algorithm and statistical table. They were whether or not the statistical table had been fully protected from an external attacker and the percentage saving in the number of primary suppressed cells used by the algorithm.

Only the results of experiments that completed were recorded. Some experiments could not complete as the mathematical solver could not handle the number of variables and/or constraints involved. The results are shown in Table 3. Algorithms E and $K5$ successfully protected 100% of the statistical tables, algorithm $K1$ successfully protected 96.5% of the statistical tables and algorithm $K3$ successfully protected only 68.2% of the statistical tables.

The average reduction in the number of primary cells requiring protection for algorithms E and $K5$ was not as large for the synthetic tables as the bootstrapped tables. This is because a large proportion of the synthetic tables are hierarchical and these contain a higher proportion of initially exposed primary suppressed cells as shown in Section 2.2. It is important that these algorithms are compared using hierarchical tables as these tend to be the type of tables that National Statistics Agencies are required to protect.

3.3. Choosing an algorithm

The time complexity of each of the components that make up the different preprocessing optimization algorithms is given in Big O notation in Table 4. From Table 4 we can see that the time complexity for each of the preprocessing optimizations is $O(cp^2)$. Therefore time complexity is not considered in the choice of algorithm.

Initially algorithms $K1$ and $K3$ are discounted as they failed to protect all the statistical tables, however they do make large savings in the number of primary suppressed cells that need to be considered when protecting a statistical table and this directly translates into savings in CPU time. If when a statistical table has failed to be protected the statistical table

Table 3
Comparison of the algorithms for finding the candidates for initially exposed primary suppressed cells.

Algorithm	Number of tables protected			Average saving (%)
	Total	Succeeded	Failed	
E	313	313	0	9.27
$K1$	313	302	11	23.64
$K3$	318	217	101	28.81
$K5$	313	313	0	11.75

Table 4
Time complexity of each of the preprocessing algorithm components. Where c is the number of constraint equations and p the number of primary cells.

Component	Time complexity
E	$O(cp^2)$
Ef	$O(cp^2)$
Ep	$O(cp^2)$
Eu	$O(cp)$
K_n	$O(cp^2)$
K_1	$O(c)$
K_{pl}	$O(cp)$
K_{nv}	$O(cp)$
K_s	$O(cp)$

Table 5
Comparison of the algorithms for finding the candidates for initially exposed primary suppressed cells.

Algorithm	Number of tables protected			Average saving (%)	Adjusted saving (%)
	Total	Succeeded	Failed		
E	313	313	0	9.27	9.27
$K1$	313	302	11	23.64	20.13
$K3$	318	217	101	28.81	−2.95
$K5$	313	313	0	11.75	11.75

is then protected using all the primary suppressed cells (i.e. no preprocessing optimization) we could still achieve an overall saving in time. To calculate the adjusted saving the extra re-run of the protection algorithm needs to be subtracted from the average saving. The adjusted results are shown in Table 5.

Even though algorithm $K1$ initially fails to protect all statistical tables it can still provide the significant saving of 20.13% when failed protections are re-run using the full set of primary cells, algorithm K3 however shows no such benefit. As $K1$ requires protection to be applied more than once we will not consider it further here as this may not be acceptable with the organizations that must implement protection. However the performance of Algorithm $K1$ implies that improvements to this preprocessing optimization are possible and that an average saving of 20% in the number of primary cells that need to be considered is achievable. Algorithm $K5$ and algorithm E both protected 100% of the statistical tables but algorithm $K5$ did so by protecting fewer of the primary suppressed cells than algorithm E. Using the Wilcoxon Signed Ranks Test we can say that we are 99.9% confident that using algorithm $K5$ will produce a greater percentage saving in the number of suppressed primary cells that need to be considered for protection than would algorithm E. The effect size, r, is 0.6594 which is above the 0.5 threshold for this to be considered a large effect. With further analysis using the Kruskal–Wallis One-Way Analysis of Variance we can be 99.9% confident in deducing the following.

1. Both algorithms produce greater savings for 2 dimension statistical tables than for 3 dimension statistical tables.
2. Both algorithms produce greater savings for 3 dimension statistical tables than for 4 dimension statistical tables.
3. Algorithm $K5$ produces greater savings for 3 and 4 dimension statistical tables than does Algorithm E.

For these reasons algorithm $K5$ was chosen as the preferred algorithm.

4. Evaluating the use of this preprocessing optimization with different cell suppression algorithms

Three different cell suppression algorithms were used to evaluate the effectiveness of this preprocessing optimization.

Kelly et al. [16] ILP: This is an Integer Linear Programming Model that guarantees to find the optimal suppression pattern.
Kelly et al. [16] LP: This is a heuristic Linear Programming Model that protects the primary suppressed cells in a statistical table one at a time.
Elimination LP: This is a heuristic Linear Programming Model that removes redundant secondary suppressed cells one at a time from an initially fully suppressed statistical table.

Implementations of each of the cell suppression algorithms were created using SAS/OR. A different set of statistical tables had to be generated for each of the cell suppression algorithms because of the differing capabilities of the algorithms. For instance the two heuristic algorithms that do not guarantee optimal suppression patterns can be used on larger statistical tables than the Kelly et al. [16] ILP which does guarantee optimal suppression patterns. Statistical tables were protected

Table 6
The percentage of statistical tables that required less CPU time with and without the preprocessing optimization.

CPU time	Kelly et al. [16] ILP (%)	Kelly et al. [16] LP (%)	Elimination LP (%)
With	71.3	90.2	91.0
Without	26.4	7.9	4.5
Equal	2.3	1.9	4.5

using these implementations with and without the preprocessing optimization (algorithm K5). The CPU time required to find the suppression patterns and the suppression pattern costs were recorded.

For each of the cell suppression algorithms the Wilcoxon Signed Ranks test indicated with $\geqslant 99.9\%$ confidence that on average using the preprocessing optimization reduced the required CPU time to find the suppression patterns. Using the preprocessing optimization however did not always reduce the required CPU time indicating that the required CPU time is influenced by more than one factor. Table 6 shows what percentage of the statistical tables required least CPU time to be protected with and without the preprocessing optimization.

An unexpected result was that for the Kelly et al. [16] LP the Wilcoxon Signed Ranks test indicated with 99.9% confidence that on average using the preprocessing optimization reduced the cost of the suppression patterns. For the 1834 statistical tables used to evaluate this preprocessing optimization on the Kelly et al. [16] LP algorithm. The test showed that for 1500 (81.8%) of the statistical tables with and without the preprocessing optimization performed identically. The difference in performance of the remaining 334 (18.2%) statistical tables can be explained by the fact that without the preprocessing optimization some *Consequentially Exposed Primary Cells* will be protected before their *Initially Exposed Primary Cells* and this might lead to extra over-protection. Out of those 334 statistical tables protecting without the preprocessing optimization performed best 74 (4.0%) times and with the preprocessing optimization performed best 260 (14.2%) times.

This preprocessing optimization method has been shown to be very effective when applied to these cell suppression algorithms. This optimization works by reducing the resources required to protect statistical tables, hence allowing statistical tables to be protected quicker or allowing larger statistical tables to be protected.

5. Conclusions

The protection of published statistical tables is a problem that more and more organizations will need to deal with. The cell suppression algorithms required to protect these statistical tables are computationally expensive. This preprocessing optimization has been shown to be very effective when applied to a range of cell suppression algorithms. This preprocessing optimization works by reducing the resources that the solver requires to protect statistical tables, hence allowing statistical tables to be protected quicker or allowing larger statistical tables to be protected. More specifically in this paper.

- A detailed description of the theory has been provided.
- A simple method for selecting *candidates for initially exposed primary suppressed cells* has been demonstrated.
- A reduction in the required CPU time to protect statistical tables has been demonstrated.
- A reduction in the suppression pattern costs has been demonstrated for one of the cell suppression algorithms.
- It has been show that this preprocessing optimization is valid for a wide variety of statistical tables. It has been shown to work with hierarchical statistical tables as well as non-hierarchical statistical tables which were considered by Serpell et al. [19].

6. Further research

This paper has also shown that there is still room for improving the algorithm for selecting *candidate initially exposed primary suppressed cells* and this requires further research.

References

[1] L. Buzzigoli, A. Giusti, An algorithm to calculate the lower and upper bounds of an array given its marginals, in: Statistical Data Protection: Proceedings of the Conference, Eurostat, Luxembourg, 1999, pp. 131–147.
[2] J. Castro, Network flows heuristics for complementary cell suppression: an empirical evaluation and extensions, in: Josep Domingo-Ferrer (Ed.), Inference Control in Statistical Databases 2002, Lecture Notes in Computer Science, vol. 2316, Springer, pp. 59–73.
[3] J. Castro, A shortest paths heuristic for statistical data protection in positive tables, INFORMS Journal on Computing 19 (4) (2007) 520–533.
[4] A. Clark, J. Smith, Improvements to Cell Suppression in Statistical Disclosure Control, End-of-Project Report ONS Contract IT-06-0960A for the Office of National Statistics (ONS), 2006.
[5] L. Cox, Disclosure risk for tabular economic data, in: P. Doyle, L.I. Lane, J.J.M. Theeuwes, L.V. Zayatz (Eds.), Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, North-Holand, 2001.
[6] L.H. Cox, Vulnerability of complementary cell suppression to intruder attack, Journal of Privacy and Confidentiality 1 (2) (2009) 235–251.
[7] P-P. de Wolfe, Hitas: a heuristic approach to cell suppression in hierarchical tables, in: Josep Domingo-Ferrer (Ed.), Inference Control in Statistical Databases, Lecture Notes in Computer Science, vol. 2316, Springer, Berlin/Heidelberg, 2002, pp. 81–98.

[8] A. Dorba, A.F. Karr, A.P. Sanil, Preserving confidentiality of high-dimensional tabulated data: statistical and computational issues, Statistics and Computing 13 (2003) 363–370.
[9] A. Dobra, S.E. Fienberg, The generalized shuttle algorithm, in: P. Gibilisco, E. Riccomagno, M.P. Rogantin, H.P. Wynn (Eds.), Algebraic and Geometric Methods in Statistics, Cambridge University Press, 2010, pp. 135–173.
[10] M. Fischetti, J.J. Salazar-González, Models and Algorithms for Optimizing Cell Suppression in Tabular Data with Linear Constraints. Technical Paper, University of La Laguna, Tenerife, 1998.
[11] M. Fischetti, J.J. Salazar-Gonzalez, Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control, Mathematical Programming 84 (2) (1999) 24–36.
[12] M. Fischetti, J.J. Salazar-González, Solving the cell suppression problem on tabular data with linear constraints, Management Science 47 (7) (2001) 1008–1027.
[13] M. Fischetti, J.J. Salazar-González, Partial cell suppression: a new methodology for statistical disclosure control, Statistics and Computing 13 (1) (2003) 13–21.
[14] A. Hundepool, A. van de Wetering, R. Ramaswamy, P. Wolf, S. Giessing, M. Fischetti, J.J. Salazar, J. Castro, P. Lowthian, Tau-ARGUS Users Manual 2007; CENEX-Project; BPA No: 769-02-TMO.
[15] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Naylor, E.S. Nordholt, G. Seri, P.-P. de Wolfe, Handbook on Statistical Disclosure Control, 2007. <http://neon.vb.cbs.nl/casc/SDCHandbook.pdf>.
[16] J.P. Kelly, B.L. Golden, A.A. Assad, Cell suppression: disclosure protection for sensitive tabular data, Networks 22 (1992) 28–55.
[17] Z. Pawlak, A. Skowron, Rudiments of rough sets, Information Sciences 177 (2007) 3–27.
[18] Z. Pawlak, A. Skowron, Rough sets: some extensions, Information Sciences 177 (2007) 28–40.
[19] M.C. Serpell, A. Clark, J. Smith, A.T. Staggemeier, Preprocessing optimization applied to the classical integer programming model for statistical disclosure control, in: Proceedings of the Conference on Privacy in Statistical Databases, 2008.
[20] M.C. Serpell, Mathematical Modelling and Artificial Intelligence Applied to Statistical Disclosure Control. PhD Thesis, University of the West of England, 2011.
[21] D. Shah, S. Zhong, Two methods for privacy preserving data mining with malicious participants, Information Sciences 177 (2007) 5468–5483.
[22] N. Shlomo, C. Young, Quality measures for statistical disclosure controlled data, in: Proceedings of the European Conference on Quality in Survey, Statistics, 2006.
[23] L. Willenborg, T. de Waal, Elements of Statistical Disclosure Control, Springer, New York, 2001.