Recent advances in optimization techniques for statistical tabular data protection

Jordi Castro

Dept. of Stat. and Operations Research
Universitat Politècnica de Catalunya
jordi.castro@upc.edu

Research Report UPC-DEIO DR 2011-03
March 2011

Recent advances in optimization techniques for statistical tabular data protection[*]

Jordi Castro
Dept. of Statistics and Operations Research
Universitat Politècnica de Catalunya
Jordi Girona 1–3
08034 Barcelona
`jordi.castro@upc.edu`
`http://www-eio.upc.es/~jcastro`

Abstract

One of the main services of National Statistical Agencies (NSAs) for the current Information Society is the dissemination of large amounts of tabular data, which is obtained from microdata by crossing one or more categorical variables. NSAs must guarantee that no confidential individual information can be obtained from the released tabular data. Several statistical disclosure control methods are available for this purpose. These methods result in large linear, mixed integer linear, or quadratic mixed integer linear optimization problems. This paper reviews some of the existing approaches, with an emphasis on two of them: cell suppression problem (CSP) and controlled tabular adjustment (CTA). CSP and CTA have concentrated most of the recent research in the tabular data protection field. The particular focus of this work is on methods and results of practical interest for end-users (mostly, NSAs). Therefore, in addition to the resulting optimization models and solution approaches, computational results comparing the main optimization techniques—both optimal and heuristic—using real-world instances are also presented.

Keywords: linear programming, network flows, mixed integer linear programming, statistical disclosure control, large-scale optimization

1 Introduction

National Statistical Agencies (NSAs) store information about individuals or *respondents* (persons, companies, etc.) in microdata files. A microdata file V of s individuals and t

[*]To appear in *European Journal of Operational Research*

	P_1	P_2	P_3	P_4	P_5	TOTAL
M_1	20	15	30	20	10	95
M_2	72	20	1	30	10	133
M_3	38	38	15	40	11	142
TOTAL	130	73	46	90	31	370

Figure 1: Two-dimensional frequency table showing number of persons for each profession and municipality.

	P_1	P_2	P_3	P_4	P_5	TOTAL
M_1	360	450	720	400	360	2290
M_2	1440	540	22	570	320	2892
M_3	722	1178	375	800	363	3438
TOTAL	2522	2168	1117	1770	1043	8620

Figure 2: Two-dimensional magnitude table showing overall salary (in 1000€) for each profession and municipality.

variables is a $s \times t$ matrix where v_{ij} is the value of variable j for individual i. Formally, it can be defined as a function

$$V : I \rightarrow D(V_1) \times D(V_2) \times \cdots \times D(V_t)$$

that maps individuals of set I to an array of t values for variables $V_1, \ldots, V_t$, $D()$ being the domain of those variables. According to this domain, variables can be classified as numerical (e.g., "age", "net profit") or categorical ("sex","economy sector"). From those microdata files, tabular data is obtained by crossing one or more categorical variables. For instance, assuming a microdata file with information of inhabitants of some region, crossing variables "profession" and "municipality" the two-dimensional *frequency* table of Figure 1 may be obtained. Instead, the table could provide information about a third variable; these tables are named *magnitude* tables. For instance, the table of Figure 2 shows the overall salary for each profession and municipality. Formally, a table is a function

$$T : D(V_{i_1}) \times D(V_{i_2}) \times \cdots \times D(V_{i_l}) \rightarrow \mathbb{R} \text{ or } \mathbb{N},$$

l being the number of categorical variables that were crossed. The result of function T (cell values) belongs to $\mathbb{N}$ for a frequency table, and to $\mathbb{R}$ for a magnitude table.

Although tabular data show aggregated information, there is a risk of disclosing individual information. For instance, if the two tables of Figures 1 and 2 are published, then any attacker knows that the salary of the unique respondent of cell (M_2, P_3) is 22000€. This is named an *external attacker*. If there were two respondents in that cell, then any of them could deduce the other's salary, becoming an *internal attacker*. Even if there was a larger number of respondents, e.g. 5, if one of them had a salary of, e.g. 18000€, there would be a disclosure risk, since the contribution of the largest respondent could exceed some predefined percentage of the cell total; this cell would be reported as sensitive by the so-called *dominance rule*. A more dangerous and difficult to protect scenario is named the *singleton problem* or *multi-attacker problem*. This happens, for instance, when two cells with a single respondent (a singleton) appear in the same table relation (e.g., a row or a column of Figures 1 and 2), such that any of them can deduce the other's contribution. This situation can be generalized to more than two singletons with collusions. The singleton problem has been discussed in Jewett (1993); Robertson (2000), and more recently in Daalmans and de Waal (2010). In all the above situations, prior to publication, NSAs have to apply some tabular data protection method. In short, those methods, whose origins date back to Bacharach (1966), basically either suppress or perturb the table cell values.

A different set of protection techniques apply directly to the original microdata files, instead of the resulting tabular data. These are out of the scope of this work. Some recent improvements on microdata protection methods can be found in Domingo-Ferrer and Mateo-Sanz (2002); Hansen and Mukherjee (2003); Muralidhar and Sarathy (2006), and in the monographs Domingo-Ferrer and Franconi (2006); Domingo-Ferrer and Magkos (2010); Domingo-Ferrer and Saigin (2008); Domingo-Ferrer and Torra (2004); Willenborg and de Waal (2000). Although the number of records in a microdata file r is in general much larger than the number of cells n in a table ($r \gg n \gg 0$), tabular data involve a number of linear constraints m, and in some real-world instances $m \gg 0$. These linear constraints model the relations between inner and total cells, the most usual relation being that the sum of some inner cells is equal to some marginal cell. Microdata protection in general involves few (if not zero) linear constrains. Therefore, tabular data protection methods rely on linear programming (LP), mixed integer linear programming (MILP), and even mixed integer quadratic programming (MIQP) technology, making the protection of complex and large tables a difficult problem. Some huge instances (which result in MILP problems of order of millions of variables and constraints) can be found in `http://www-eio.upc.es/~jcastro/data.html`.

The detection of the sensitive cells to be protected is made by applying some sensitivity rules. Although it is an important step of the data protection process, it is not covered here. Indeed, currently, those rules do not rely on optimization or operations research methodology. Practical details about these rules can be found in Hundepool et al. (2010). Additional information can be found in Domingo-Ferrer and Torra (2002); Robertson and Ethier (2002).

Although it contains references to recent literature, this paper is not meant to be a comprehensive survey on statistical disclosure control of tabular data. The interested reader is referred, for instance, to the research monographs Domingo-Ferrer and Franconi (2006); Domingo-Ferrer and Magkos (2010); Domingo-Ferrer and Saigin (2008); Domingo-Ferrer and Torra (2004); Willenborg and de Waal (2000), and the recent survey Salazar-González (2008). Guidelines to end-users of tabular data protection methods can be found in the handbook Hundepool et al. (2010). Compared to those previous works, the main contributions of this paper are: (i) it focuses on two of today more relevant techniques for NSAs (namely, cell suppression problem, and controlled tabular adjustment); (ii) not only the resulting optimization models are presented, but also the main solution techniques are sketched; (iii) it reports computational results comparing the available techniques using state-of-the-art software for tabular data protection, showing the lacks and benefits of the different models and solution approaches.

The structure of the paper is as follows. Section 2 shows the different types of tables that can be obtained, and how they are modeled; this background is needed for the subsequent sections. Section 3 introduces tabular data protection methods. Sections 4 and 5 focus on two of the most widely used protection techniques, the *cell suppression* and the *controlled tabular adjustment*, both describing the optimization models and outlining the main solution approaches.

2 Tabular data: types and models

Some protection methods of Section 3 are either only valid or may be specialized (i.e., made more efficient) for some particular type of tabular data. It is thus instrumental to know in advance the type of table to be protected and how to model it.

2.1 Classification of tables

Tables can be classified according to different criteria. Two of the simplest criteria for classification are "cell values" and "sign of cell values". According to the cell values, the two classes of tables were already introduced in Section 1: *frequency tables*—also named contingency tables—, and *magnitude tables.* According to the sign of cell values, tables are classified as either *positive* or *general* tables. Cell values of positive tables are non-negative, which is the most usual situation. For instance, all frequency tables and most magnitude tables, like "salary" for "profession"$\times$"municipality", are positive tables. Cell values of general tables can be positive or negative. An example of a general table would be "variation of gross domestic product" for "year"$\times$"state". Assuming a table is general instead of positive can be instrumental in some protection methods. Indeed, those methods usually involve the solution of difficult LP or MILP problems; the lower bounds of some variables are $-\infty$ for general tables (0 for positive ones). This property has been exploited in some efficient heuristics for general tables (Carvalho et al., 1994).

For a modelling point of view, and to exploit the type of table in the resulting LP or MILP from the data protection method, the most important classification criteria is "table structure". Indeed, some protection methods can only be applied to particular table structures. Moreover, the different models in Subsection 2.2 are tailored for some table structures. According to their structure, tables may be classified as *single k-dimensional*, *hierarchical*, or *linked* tables. A single k-dimensional table is obtained by crossing k categorical variables. For instance, tables of Figures 1–2 are two-dimensional. A hierarchical table is a set of tables obtained by crossing some variables, and a number of these variables have a hierarchical relation. For instance, consider the three tables of Figure 3. The left subtable shows number of respondents for "region"$\times$"profession"; the middle subtable, a "zoom in" of region R_2, provides the number of respondents for "municipality"(of region R_2)$\times$"profession"; finally the right subtable, "zip code"$\times$"profession", details municipality R_{21}. This table belongs to a particular class named 1H2D, two-dimensional tables with one hierarchical variable. Finally, linked tables are the most general situation. A linked table is a set of tables obtained from the same microdata file. In theory, the set of all tables obtained from a microdata file should be considered together as a (likely huge) linked table. Hierarchical and k-dimensional tables are particular cases of linked tables. Note that, in theory, the only safe way for protecting all

	C_1	C_2	C_3
R_1	5	6	11
R_2	10	15	25
R_3	15	21	36

T_1

	C_1	C_2	C_3
R_{21}	8	10	18
R_{22}	2	5	7
R_2	10	15	25

T_2

	C_1	C_2	C_3
R_{211}	6	6	12
R_{212}	2	4	6
R_{21}	8	10	18

T_3

Figure 3: Hierarchical table made of three subtables: "region"×"profession", "municipality"×"profession" and "zip code"×"profession".

a_{11}	...	a_{1c}	$a_{1(c+1)}$
...	...	...	...
a_{r1}	...	a_{rc}	$a_{r(c+1)}$
$a_{(r+1)1}$	...	$a_{(r+1)c}$	$a_{(r+1)(c+1)}$

Figure 4: General two-dimensional table.

the tables from a microfile, is to jointly protect them as a single linked table. Unfortunately, in many cases the size of the resulting table would be excessive for current LP or MILP technology.

2.2 Modelling tables

Linked tables are the more general case, and a model for them is valid for the other types of tables. However, below are outlined particular models for two-dimensional, three-dimensional, 1H2D tables, and, finally, a general model for linked tables, valid for any table.

Two-dimensional tables. A two-dimensional table of $r+1$ rows and $c+1$ columns as the one of Figure 4 is modeled by constraints

$$\begin{aligned} \sum_{j=1}^{c} a_{ij} &= a_{i(c+1)} \quad i=1,\ldots,r \\ \sum_{i=1}^{r} a_{ij} &= a_{(r+1)j} \quad j=1,\ldots,c. \end{aligned} \tag{1}$$

Constraints (1) can be represented by the bipartite network of Figure 5. This allows the application of efficient network optimization algorithms, such as those for minimum cost network flows, or shortest-paths (Ahuja et al., 1993). This fact was originally noticed in Bacharach (1966), and it has been extensively used in other works (Carvalho et al., 1994; Castro, 2002, 2004, 2007a; Cox, 1995; Fischetti and Salazar-González, 1999; Kelly et al., 1992).

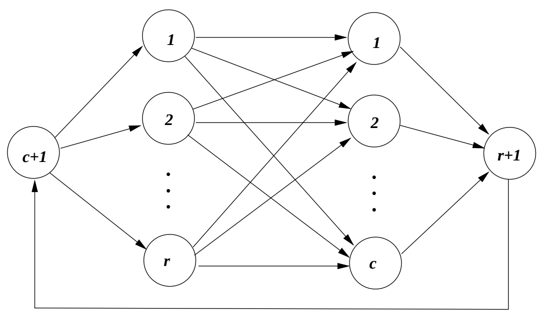

Figure 5: Network representing constraints (1).

Three-dimensional tables. The linear constraints of a three-dimensional table of $r+1$ rows, $c+1$ columns and $l+1$ levels (levels refer to categories of third variable) are

$$\begin{aligned} \sum_{i=1}^{r} a_{ijk} &= a_{(r+1)jk} \quad j = 1 \ldots c, \quad k = 1 \ldots l \\ \sum_{j=1}^{c} a_{ijk} &= a_{i(c+1)k} \quad i = 1 \ldots r, \quad k = 1 \ldots l \\ \sum_{k=1}^{l} a_{ijk} &= a_{ij(l+1)} \quad i = 1 \ldots r, \quad j = 1 \ldots c. \end{aligned} \tag{2}$$

Note the above constraints correspond to a cube of data. Rearranging (2), these constraints can be modeled as a multicommodity network (Castro, 2002). Variables $x_{ijk}, i = 1, \ldots, r, j = 1, \ldots, c, k = 1, \ldots, l$ are ordered according to k, i.e., $x = (x_{ij1}^T, \ldots, x_{ijl}^T)^T$. Each group for a particular k contains rc variables, and it corresponds to a layer of the cube of data. Each layer is a two-dimensional table, which is modeled as the network of Figure 5. Data for each particular layer (or level) corresponds to a commodity. The l commodities are linked by capacity constraints, forcing that the sum for all the commodities (levels) is equal to the marginal level. The resulting constraints matrix structure is

$$A = \begin{array}{c|cccc|l} & a_{ij1} & a_{ij2} & \ldots & a_{ijl} & \\ \hline & N & & & & \text{for } k=1 \\ & & N & & & \text{for } k=2 \\ & & & \ddots & & \quad \vdots \\ & & & & N & \text{for } k=l \\ & I & I & \ldots & I & \text{linking constraints,} \\ \hline \end{array} \tag{3}$$

N being the node-arc incidence network matrix for the two-dimensional tables of each level, and $I \in \mathbb{R}^{rc \times rc}$ being the identity matrix. Exploiting this structure, significant computational savings can be obtained (Castro, 2005, 2007b).

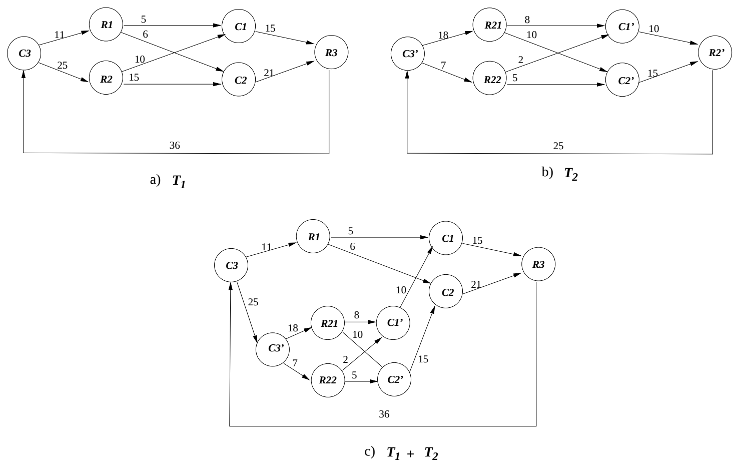

Figure 6: Intermediate network representing 1H2D table of Figure 3 (first iteration).

Hierarchical tables. In general, hierarchical tables have to be modeled as a general linked table. However, for the particular case of 1H2D tables, as that of Figure 3, it is possible to obtain a network representation. In short, the algorithm for building the network of a 1H2D table consists of the following stages (Castro, 2007a):

1. Build a tree of subtables representing the structure of the 1H2D (i.e., for table of Figure 3, the root node would be the left table; the middle table would be a descendant of the root table; and the right table would be a descendant of the middle table).
2. Perform a search on the tree of subtables, using, for instance, a breadth-first-search, and build the breadth-first-list.
3. Build the network for each subtable.
4. For all the subtables in the breadth-first-list, embed the network of a table within the table of its parent table.

The above procedure is done in linear time. For instance, for the 1H2D table of Figure 3 after the first iteration the network of Figure 6 is obtained; after the second and last iteration the definitive network of Figure 7 would be obtained. This network model was successfully used for a fast heuristic for protection of 1H2D tables in Castro (2007a).

Linked tables. In general, any table can be modeled as a set of n cells $a_i, i = 1, \dots, n$, which satisfy a set of m linear relations $Aa = b$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. In practice cell values must be between some lower and upper bounds $l_i, u_i, i = 1, \dots, n$. If the table is positive then $l_i \geq 0, i = 1, \dots, n$. Each row of matrix $A = (a_{ij}), i = 1, \dots, m, j = 1, \dots, n$ is related to a table linear relation, and $a_{ij} \in \{1, 0, -1\}$. The value -1 of every equation is related to the marginal or total cell. The above types of tables are particular cases where A is either a node-arc incidence network flows matrix, or a multicommodity network flows matrix. In

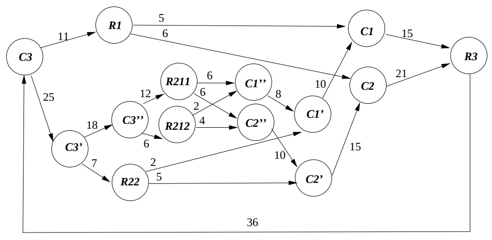
Figure 7: Final network representing 1H2D table of Figure 3 (second iteration).

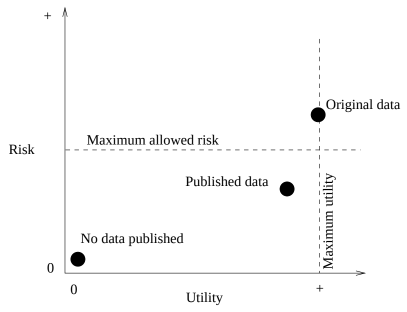

Figure 8: The risk-utility graph.

real-world problems the dimension of n and m can be very large (order of millions of variables and constraints).

3 Tabular data protection methods

The purpose of data protection methods is to reduce the disclosure risk of confidential information, while preserving the utility of the released data (where data utility means the value of a given data release as an analytical resource). Disclosure risk is usually reduced by modifying or hiding some information. Note that disclosure risk can only be reduced, not completely avoided, unless no data is published. This is clearly shown in the risk-utility graph of Figure 8. The goal is to publish tabular data as close as possible to the original data, i.e., with a similar utility, but below the maximum acceptable risk. Of course, this depends on how risk and utility are modeled, and every protection method has a particular model (all of them equally valid) within the resulting optimization problem.

Broadly, tabular data protection methods can be classified as:

- Non-perturbative: they don't change the original data, instead they suppress data or change the table structure, such as, for instance, *recoding* and *cell suppression*.
- Perturbative: they provide an alternative table with modified values. *Controlled rounding* and *controlled tabular adjustment* belong to this class.

Original table

	P_1	P_2	P_3	P_4	P_5	TOTAL
M_1	20	15	30	20	10	95
M_2	72	20	**1**	30	10	133
M_3	38	38	15	40	11	142
TOTAL	130	73	46	90	31	370

Recoded table

	P_1	$P_2 + P_3$	P_4	P_5	TOTAL
M_1	20	45	20	10	95
M_2	72	**21**	30	10	133
M_3	38	53	40	11	142
TOTAL	130	119	90	31	370

Figure 9: Original and recoded table after aggregation of professions P_2 and P_3.

Cell suppression and controlled tabular adjustment have concentrated most of the recent research in the tabular data protection field. Below sections 4 and 5 are devoted to these particular techniques. Of the remaining approaches, recoding is likely the simplest one. It consists in aggregating or changing some of the categorical variables that define the table, to satisfy the sensitivity rules. This is shown in the example of Figure 9, whose tables report the number of respondents for "profession" $\times$ "municipality". The main advantages of this approach are its simplicity and that it does not require any sophisticated algorithm. The main inconveniences are that it changes the table structure, that an excessive aggregation may significantly reduce the utility of the resulting table, and that it may have a high disclosure risk; indeed, if a variety of tables with recoded versions of the same categorical variable (like "size class") are published, it becomes very likely that by suitable differencing, some cell entries can be disclosed.

An alternative approach is *rounding*, which achieves protection by rounding all cell tables to a multiple of a certain base number r. Figure 10 shows an example of a two-dimensional table using a base number $r = 5$. Note that the total cell could not be rounded to the closest multiple of 5, otherwise the resulting table would not be additive. This variant that guarantees additivity is named *controlled rounding*. Although controlled rounding was already in use two decades ago (Cox and George, 1989), some recent extensions using lower and upper protection levels have been considered (Salazar-González, 2006). The complexity of the resulting model is similar to that of cell suppression, resulting in a large MILP which is solved by Benders decomposition (Benders, 2005). One of the main drawbacks of controlled rounding is that it forces deviations for all the cells that are not originally a multiple of the base r, reducing the utility of the resulting table. In addition, to guarantee additivity, total cells have also to be rounded, likely to a multiple which can be far from the original value. The controlled tabular adjustment approach of Section 5, which also perturbs cell values, avoids some of these inconveniences, and it may guarantee a greater utility of the resulting modified table.

4 Cell suppression

Given a set of sensitive cells to be protected (named *primary* cells), the purpose of the *cell suppression problem* (CSP) is to find a set of additional cells (named *secondary* or *complementary* cells) that guarantees that the value of primary cells can not be guessed, and minimize some information loss criteria. Figure 11 shows an example of a two-dimensional

Original table

	P_1	P_2	P_3	TOTAL
M_1	20	24	28	72
M_2	38	38	40	116
M_3	40	39	42	121
TOTAL	98	101	110	**309**

Rounded table

	P_1	P_2	P_3	TOTAL
M_1	20	25	30	75
M_2	40	40	40	120
M_3	40	40	40	120
TOTAL	100	105	110	**315**

Figure 10: Original and rounded table using a base number $r = 5$.

Original table

	P_1	P_2	P_3	TOTAL
M_1	20	24	28	72
M_2	38	38	**40**	116
M_3	40	39	42	121
TOTAL	98	101	110	309

Protected table

	P_1	P_2	P_3	TOTAL
M_1		24		72
M_2		38		116
M_3	40	39	42	121
TOTAL	98	101	110	309

Figure 11: Original table with primary cell in boldface, and protected table after suppression of three secondary cells.

table with only one primary cell in boldface. If this was the only removed cell, its value could be retrieved from marginals. Therefore, the suppression of three additional complementary cells is needed. From the protected table of Figure 11, any attacker may deduce a lower and upper bound for the primary cell. Indeed, considering variables x_{11}, x_{13}, x_{21}, x_{23} for the primary and secondary cells, a lower bound $\underline{a_{23}}$ and an upper bound $\overline{a_{23}}$ for the primary cell can be obtained by solving

$$
\begin{array}{rl}
\underline{a_{23}} = \min & x_{23} \\
\text{s. to} & x_{11} + x_{13} = 72 - 24 \\
 & x_{21} + x_{23} = 116 - 38 \\
 & x_{11} + x_{21} = 98 - 40 \\
 & x_{13} + x_{23} = 110 - 42 \\
 & (x_{11}, x_{13}, x_{21}, x_{23}) \geq 0
\end{array}
\quad \text{and} \quad
\begin{array}{rl}
\overline{a_{23}} = \max & x_{23} \\
\text{s. to} & x_{11} + x_{13} = 72 - 24 \\
 & x_{21} + x_{23} = 116 - 38 \\
 & x_{11} + x_{21} = 98 - 40 \\
 & x_{13} + x_{23} = 110 - 42 \\
 & (x_{11}, x_{13}, x_{21}, x_{23}) \geq 0.
\end{array}
\tag{4}
$$

The solutions to (4) are $\underline{a_{23}} = 20$ and $\overline{a_{23}} = 68$. If, for instance, *lower* and *upper protection levels* of $lpl = upl = 10$ were imposed (i.e., the protection pattern must guarantee that no attacker can deduce a value of the sensitive cell within the range $[40 - lpl, 40 + upl] = [30, 50]$), then this cell would be protected by this suppression pattern since $\underline{a_{23}} = 20 < 30$ and $\overline{a_{23}} = 68 > 50$.

The above example illustrated the basics of CSP. In general, any instance of CSP is defined by the following parameters:

- A general table $a_i, i = 1, \ldots, n$, with m linear relations $Aa = b$, $a = (a_1, \ldots, a_n)^T$ being the vector of cell values.

- Upper and lower bounds u and l for the cell values, which are assumed to be known by any attacker: $l \leq a \leq u$ (e.g., $l = 0$, $u = +\infty$ for a positive table).
- Vector of nonnegative weights associated to the cell suppressions $w_i, i = 1, \dots, n$, i.e., w_i measures the cost (or data utility loss) associated to the suppression of cell i. The objective function to be minimized by CSP is the sum of the weights of suppressed cells. Then, if $w_i = 1$ the number of suppressed cells is minimized; if $w_i = a_i$ the overall value suppressed is minimized.
- Set $\mathcal{P} \subseteq \{1, \dots, n\}$ of primary or sensitive cells, decided in advance by applying some sensitivity rules.
- Lower and upper protection levels for each primary cell lpl_p and upl_p $p \in \mathcal{P}$ (usually either a fraction of a_p or directly obtained from the sensitivity rules).

CSP looks for a set $\mathcal{S}$ of secondary cells to be removed such that for all $p \in \mathcal{P}$

$$\underline{a_p} \leq a_p - lpl_p \quad \text{and} \quad \overline{a_p} \geq a_p + upl_p, \tag{5}$$

$\underline{a_p}$ and $\overline{a_p}$ being defined as

$$\begin{array}{lll} \underline{a_p} = & \min & x_p \\ & \text{s. to} & Ax = b \\ & & l_i \leq x_i \leq u_i \quad i \in \mathcal{P} \cup \mathcal{S} \\ & & x_i = a_i \quad i \notin \mathcal{P} \cup \mathcal{S} \end{array} \quad \text{and} \quad \begin{array}{lll} \overline{a_p} = & \max & x_p \\ & \text{s. to} & Ax = b \\ & & l_i \leq x_i \leq u_i \quad i \in \mathcal{P} \cup \mathcal{S} \\ & & x_i = a_i \quad i \notin \mathcal{P} \cup \mathcal{S}. \end{array} \tag{6}$$

The classical model for CSP was originally formulated in Kelly et al. (1992). It considers two sets of variables: (1) $y_i \in \{0, 1\}, i = 1, \dots, n$, is 1 if cell i has to be suppressed, 0 otherwise; (2) for each primary cell $p \in \mathcal{P}$, two auxiliary vectors $x^{l,p} \in \mathbb{R}^n$ and $x^{u,p} \in \mathbb{R}^n$, which represent cell deviations (positive or negative) from the original a_i values. The resulting model is

$$\begin{array}{ll} \min & \displaystyle\sum_{i=1}^{n} w_i y_i \\ \text{s. to} & \left. \begin{array}{rlll} & Ax^{l,p} & = 0 & \\ (l_i - a_i) y_i \leq & x_i^{l,p} & \leq (u_i - a_i) y_i & i = 1, \dots, n \\ & x_p^{l,p} & \leq -lpl_p & \\ & & & \\ & Ax^{u,p} & = 0 & \\ (l_i - a_i) y_i \leq & x_i^{u,p} & \leq (u_i - a_i) y_i & i = 1, \dots, n \\ & x_p^{u,p} & \geq upl_p & \end{array} \right\} \forall\, p \in \mathcal{P} \\ & y_i \in \{0, 1\} \quad i = 1, \dots, n. \end{array} \tag{7}$$

The inequality constraints of (7) with both right- and left-hand sides impose bounds on $x_i^{l,p}$ and $x_i^{u,p}$ when $y_i = 1$, and prevent deviations in non-suppressed cells (i.e., $y_i = 0$). Clearly, the constraints of (7) guarantee that the solutions of the linear programs (6) will satisfy (5).

Note that (7) gives rise to a MILP problem of n binary variables, $2n|\mathcal{P}|$ continuous variables, and $2(m+2n)|\mathcal{P}|$ constraints. This problem is very large even for tables of moderate size and number of primary cells. For instance, for a table of 8000 cells, 800 primaries, and 4000 linear relations, we obtain a MILP with 8000 binary variables, 12,800,000 continuous variables, and 32,000,000 constraints. However, (7) is the basis of several solution methods, either optimal or heuristic, to be discussed below. Subsection 4.4 performs a computational comparison of these approaches using public and private real-world instances.

The classical CSP model (7) does not protect against the singleton problem described in Section 1. We outline below two possible extensions of (7) for the singleton problem.

- The first approach provides a "local" protection since it only focuses on individual table relations with singletons. This approach is simple, but since inter-relations are not considered it may offer a weak protection (although higher than the standard model (7)). Denoting by $\mathcal{G}_j$ and $\mathcal{H}_j, j = 1, \ldots, m$, the set of singletons (i.e., cells with one respondent) and set of cells in table relation j, respectively, we can ensure than at least one additional cell, different from those in $\mathcal{G}_j$, has to be suppressed by adding to (7) the following set of constraints:
$$\sum_{i \in \mathcal{H}_j} y_i \geq |\mathcal{G}_j| + 1 \qquad j = 1, \ldots, m. \tag{8}$$
- The second approach would provide a more "global" protection, resulting in a model which is still practical from a computational point of view (unlike, for instance, the theoretical "global" approach of Daalmans and de Waal (2010)). It deals with all the table relations, considering the (perhaps unrealistic) worst-case scenario where all the singletons may collude to recompute some non-singleton sensitive cell. This approach works as follows. When computing the lower and upper bounds $\underline{a_p}$ and $\overline{a_p}$ through (6), which are available to any attacker, we assume that the cell values of singletons are known, although they are sensitive (i.e., we compute $\underline{a_p}$ and $\overline{a_p}$ by assuming collusion of singletons). For this purpose, denoting by $\mathcal{G}$ the set of singletons, the third set of constraints of the minimization and maximization problems of (6) should be replaced by
$$x_i = a_i \quad i \notin ((\mathcal{P} \setminus \mathcal{G}) \cup \mathcal{S} \cup \{p\}), \tag{9}$$
i.e., we consider the attacker knows the values of single-respondent cells, excluding cell p when $p \in \mathcal{G}$; if $p \in \mathcal{G}$ then (6) with the new constraints (9) provides the lower and upper bounds of p when the other singletons collude. This means that the constraints of (7)
$$\begin{array}{rcll} (l_i - a_i)y_i \leq & x_i^{l,p} & \leq (u_i - a_i)y_i & \quad i = 1, \ldots, n \\ (l_i - a_i)y_i \leq & x_i^{u,p} & \leq (u_i - a_i)y_i & \quad i = 1, \ldots, n \end{array}$$
should be replaced by the new ones
$$\begin{array}{rcll} (l_i - a_i)y_i \leq & x_i^{l,p} & \leq (u_i - a_i)y_i & \quad i \in (\{1, \ldots, n\} \setminus \mathcal{G}) \cup \{p\} \\ & x_i^{l,p} & = 0 & \quad i \in \mathcal{G} \setminus \{p\} \\ (l_i - a_i)y_i \leq & x_i^{u,p} & \leq (u_i - a_i)y_i & \quad i \in (\{1, \ldots, n\} \setminus \mathcal{G}) \cup \{p\} \\ & x_i^{u,p} & = 0 & \quad i \in \mathcal{G} \setminus \{p\}. \end{array} \tag{10}$$

It is worth noting that, by assuming this worst-case scenario where all the singletons collude, this procedure may cause oversuppression—the information loss can thus be excessive—, resulting in an unacceptable table for practitioners.

4.1 Benders decomposition for CSP

Problem (7) can be approached by means of a Benders decomposition (Benders, 2005). Initially applied to two-dimensional tables (Fischetti and Salazar-González, 1999), it was later extended to general tables (Fischetti and Salazar-González, 2001), and other CSP variants (Salazar-González, 2004). Benders algorithm is a cut generation procedure that iteratively solves a master problem in variables $y_i \in \{0,1\}, i = 1,\dots,n$—which provides a suppression pattern—, and $|\mathcal{P}|$ subproblems (one per primary cell) which "inform" about the protection provided by this pattern to primary cells. If all primaries are protected, then the suppression pattern is optimal. Otherwise, a violated "protection cut" is added to the master problem, and the master is solved again. Iterations are performed until the optimal solution is found (which is guaranteed, since the number of iterations is finite). Broadly, the Benders decomposition algorithm applied to CSP is as follows:

1. Initializations: set initial set of protection constraints $\mathcal{J} = \emptyset$.
2. Solve master problem:

$$\begin{array}{ll} \min & \sum_{i=1}^{n} w_i y_i \\ \text{s. to} & y_p = 1 \;\; \forall p \in \mathcal{P} \\ & y_i \in \{0,1\} \;\; i = 1,\dots,n \\ & {v^j}^T y \geq \beta^j \quad j \in \mathcal{J}, \end{array} \tag{11}$$

   where $v^j \in \mathbb{R}^n$ and $\beta^j \in \mathbb{R}$ are the left and right hand sides of protection cuts (to be defined in below steps 3–4). Note primary cells are always suppressed, even for $\mathcal{J} = \emptyset$.
3. Check whether suppression pattern $y_i, i = 1,\dots,n$ satisfies lower protection level lpl_p for each cell $p \in \mathcal{P}$:

   - For each primary cell $p \in \mathcal{P}$ it has to be checked whether deviations x (supraindices ^{l,p} are suppressed to simplify the notation) that satisfy the first group of constraints of (7)

$$\begin{array}{rll} & Ax & = 0 \\ (l_i - a_i)y_i \leq & x_i & \leq (u_i - a_i)y_i \qquad i = 1,\dots,n \\ & x_p & \leq -lpl_p \end{array}$$

     exist, or equivalently that

$$\begin{array}{rlll} -lpl_p \geq \min & x_p & & \\ \text{s. to} & Ax = 0 & & [\lambda] \\ & x_i \geq (l_i - a_i)y_i & i = 1,\dots,n & [\mu_l] \\ & x_i \leq (u_i - a_i)y_i & i = 1,\dots,n & [\mu_u], \end{array} \tag{12}$$

     λ, μ_l and μ_u being the set of Lagrange multipliers (also known as dual variables) of each group of constraints.

- Problem (12) has always a solution: (i) it is feasible, since $x = 0$ (no deviation), which corresponds to the original table is feasible (but not optimum); (ii) it is not unbounded, since $x_p \geq l_p - a_p > -\infty$ (e.g., if table is positive then $l_p = 0$).
- By LP duality, the dual of (12) is

$$\begin{array}{ll} \max & 0\lambda + \sum_{i=1}^{n}(l_i - a_i)y_i\mu_{l_i} - \sum_{i=1}^{n}(u_i - a_i)y_i\mu_{u_i} = \\ & = \sum_{i=1}^{n}\left((l_i - a_i)\mu_{l_i} - (u_i - a_i)\mu_{u_i}\right)y_i \\ \text{s. to} & A^T\lambda + \mu_l - \mu_u = e_p \\ & \mu_l \geq 0, \quad \mu_u \geq 0, \end{array} \tag{13}$$

  where e_p is the p-th column of the identity matrix.
- Then, lower protection level of primary cell p is satisfied if

$$-lpl_p \geq \sum_{i=1}^{n}\left((l_i - a_i)\mu_{l_i} - (u_i - a_i)\mu_{u_i}\right)y_i, \tag{14}$$

  μ_l and μ_u being the solution of (13).
- If (14) holds for all $p \in \mathcal{P}$, then the suppression pattern y guarantees lower protection levels. If, for some $p \in \mathcal{P}$, (14) is not satisfied, then it is added to $\mathcal{J}$, the set of protection constraints of the master problem.

4. Check whether suppression pattern $y_i, i = 1, \ldots, n$ satisfies upper protection level upl_p for each cell $p \in \mathcal{P}$:

   - We proceed as in the previous case for lower protection levels. If the protection of some primary is violated, a protection constraint (similar to (14)) is added to the master problem:

$$upl_p \leq \sum_{i=1}^{n}\left(-(l_i - a_i)\mu_{l_i} + (u_i - a_i)\mu_{u_i}\right)y_i \tag{15}$$

     μ_l and μ_u being the solutions of the dual problem for the upper protection.

5. If at steps 3–4 no protection constraint was violated, then the current suppression pattern y_i is optimal. Otherwise we go to step 2 to solve the master problem with an updated set $\mathcal{J}$.

As in other applications of Benders decomposition, the performance of the method depends of the number of iterations to be performed. In general, it has been observed that this approach is very efficient for CSP instances in two-dimensional tables. However, for medium-large sized and complex tables, this approach becomes computationally expensive. On the other hand, compared to below alternative approaches, it guarantees an optimal solution if large enough CPU times are provided (unfortunately, this is not always allowed in real practice).

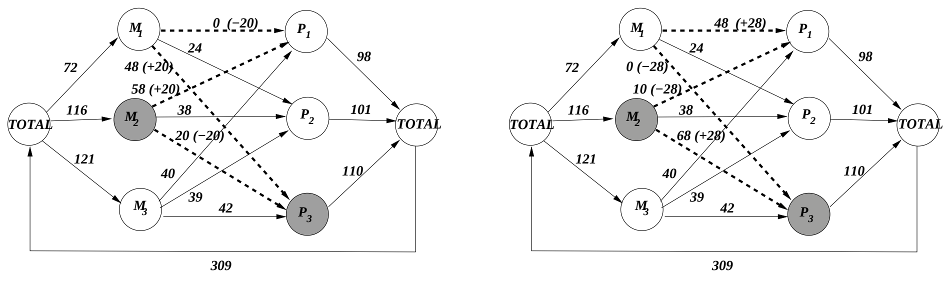

Figure 12: Solutions of (4) (left figure for $\underline{a_{23}}$, right figure for $\overline{a_{23}}$) as minimum cost network flow problems. Values in brackets are the cell increments and decrements.

The above Benders algorithm for CSP can be easily extended with the two approaches suggested in Section 4 for the singleton problem. For the first "local" protection approach, the set of constraints (8) should be added to the master problem (11) while the subproblems (12) and its equivalent for the upper protection would not change. In the second "global" protection approach, the new constraints (10) should replace the last two sets of constraints of the subproblems (12) and its equivalent for the upper protection, while the master problem (11) would not change.

4.2 Network optimization heuristics for CSP

Most heuristic approaches for (7) find a feasible, hopefully good point, by network optimization algorithms (in particular, minimum cost network flows, and shortest paths (Ahuja et al., 1993)). Unfortunately, those heuristics can only be used in tables that accept a network representation: two-dimensional and 1H2D hierarchical tables (the latter is however an interesting case for NSAs). Some attempts have been made for extending them to three-dimensional tables (Dellaert and Luijten, 1999), but as mentioned in Section 2.2, three-dimensional tables correspond to multicommodity flows, and therefore "standard single-commodity" network optimization procedures are not valid (and rather unsuccessful). Among those heuristics we find the seminal paper Kelly et al. (1992), and Castro (2002) and Cox (1995), which rely on minimum-network cost flows. For general tables Carvalho et al. (1994) suggested an efficient procedure based on shortest paths. Some of those ideas were sensibly combined in the approach of Castro (2007a), based on shortest paths but valid for positive tables. This approach is very efficient, but it can only be applied to either two-dimensional or 1H2D hierarchical tables.

Figure 12 illustrates how network flows can be used for cell suppression. The network of Figure 12 represents the table of the example of Figure 11. For each cell a_i an edge $x_i = (s, t)$ is depicted (s and t are the source and target node for each cell), and two arcs are considered: the forward arc $x_i^+ = (s, t)$ for the cell value increments, and the backward arc $x_i^- = (t, s)$ for the cell value decrements. The two optimization problems (4) are solved as two minimum cost network flow problems, by finding the maximum and minimum flow that can be sent through the arcs associated to suppressed cells. Those flows are 20 and 68, which match the

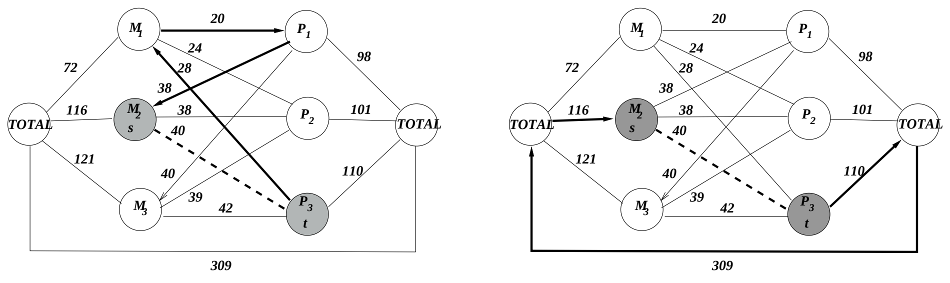

Figure 13: Protections provided to cell $(2,3)$ by two different paths: either using internal cells (left figure), or total cells (right figure).

solutions of (4), and since they are out of the protection interval $[40 - lpl, 40 + upl] = [30, 50]$ then cell $(2,3)$ is protected. If this procedure is repeated for all primary cell, and the values are out of the respective protection intervals, then a feasible solution is available for CSP. This idea is the basis of network flows heuristics for CSP. The general solution scheme is as follows:

1. Build the network associated to either the two-dimensional or 1H2D table.
2. For each cell a_i two arcs $x_i^+ = (s,t)$ and $x_i^- = (t,s)$ are considered, related to cell value increments and decrements.
3. A (hopefully good) feasible solution to CSP is obtained by solving two minimum cost network flow problems for each primary cell $p \in \mathcal{P}$:
   - Problem for upper protection level upl_p. Considering forward arc $x_p^+ = (s,t)$, send upl_p flow units from t to s excluding arc $x_p^- = (t,s)$. Cells in the resulting cycle are suppressed (they are either cells in $\mathcal{P}$ or in $\mathcal{S}$—set of secondary cells).
   - Problem for lower protection level lpl_p. Considering backward arc $x_p^- = (t,s)$: send lpl_p flow units from s to t excluding arc $x_p^+ = (s,t)$. As before, cells in the resulting cycle are suppressed.

   When solving the minimum cost flows problems, cells in $\mathcal{P}$ or previously suppressed cells in $\mathcal{S}$ are preferred. This is achieved by using appropriate arc costs.
4. Once all primary cells have been considered, the sets $\mathcal{P}$ and $\mathcal{S}$ provide a feasible (suboptimal) suppression pattern.

The main drawback of the previous approach is that up to $2|\mathcal{P}|$ minimum cost network flow problems have to be solved. Although minimum cost network flows problems are more efficient than LP problems, for large instances they may require a significant amount of time.

A recent improvement (Castro, 2007a) was obtained by replacing the minimum cost network flow problems by shortest paths ones. Figure 13 illustrates this approach, using the same example of Figure 12. Left picture of Figure 13 shows the shortest path (for some particular arc costs) from t to s for the primary cell $a_p = (s,t)$. Cells of arcs in the shortest path are the ones to be suppressed (marked with thick lines). The minimum of the cell values for the cells associated to forward arcs in the shortest paths (this minimum will be denoted as γ^+) provides the maximum decrement of cell p that preserves the nonnegativity of cells in the cycle. In this example $\gamma^+ = \min\{40, 20\} = 20$. Similarly, the minimum of the cell values for the cells associated to backward arcs in the shortest path (denoted as γ^-) provides the maximum increment that preserves the nonnegativity of cells in the cycle. In this case $\gamma^- = \min\{28, 38\} = 28$. Therefore, γ^+ and γ^- are the lower and upper protections given by this shortest path to primary cell p. The right picture of Figure 13 shows another shortest path, for some other particular arc costs, only involving total cells. In this case $\gamma^+ = \min\{40, 110, 309, 116\} = 40$ (minimum of forward arcs) and $\gamma^- = +\infty$ (there is no backward arc), which means that the maximum decrement of this cell is 40 (i.e., lower protection level is 40), and it can be infinitely increased (i.e., upper protection level is $+\infty$). This situation in practice is avoided, since total cells usually don't want to be suppressed and high costs are set to them. If some lower or upper protection level for some primary is not satisfied, additional shortest paths are computed. This procedure is repeated for all primary cell. The set of cells associated to arcs of any shortest path is the set of complementary cells to be suppressed. Figure 14 outlines the overall procedure, where $\mathcal{S}$ is the set of secondary cells, *Clpl* and *Cupl* are vectors that keep the current protection achieved by each primary cell with previously computed shortest paths, and $\mathcal{TT}$ is the set of cells already used in shortest paths for protecting the current primary cell p. This heuristic is very fast in practice, since it only requires the solution of shortest paths algorithms. The quality of the upper bound provided for the optimal objective function depends of the particular instance. Some LP models have been used for obtaining a lower bound (Kelly et al., 1992; Castro, 2007a).

It is worth noting that a solution that overcomes the singleton problem can be easily obtained with either network flows or shortest paths heuristics as follows. As seen above, for each sensitive cell p either two network flows problems or a sequence of shortest paths problems has to be solved. The arcs in the cycle or in the shortest path are used to compute the upper and lower protection levels for the sensitive cell p. We just have to forbid singletons in either these cycles or shortest paths, such that the protection to any sensitive cell is provided by other non-singleton cells. This can be done by imposing large costs to the arcs associated to singletons in the network flows or shortest paths problems to be solved.

4.3 Other heuristics for CSP

Two other heuristics are finally mentioned, currently being used by NSAs. Indeed, these two heuristics were developed at NSAs to overcome the drawbacks of the optimal approach of Subsection 4.1 (namely, computationally too expensive for large instances) and the heuristic of Subsection 4.2 (namely, not applicable to general linked tables). The *hypercube* (Giessing and Repsilber, 2002), initially developed for k-dimensional tables, is a simple and fast procedure. For two-dimensional tables, given a primary cell a_{ij} (row i, column j), it finds a protection pattern of three cells $a_{il} - a_{hl} - a_{hj}$ (a rectangle, or two-dimensional cube). For a three-dimensional table, it would find a pattern of seven cells (three-dimensional cube). It can thus be seen as a network flows approach that only considers a subset of the flows (thus providing less quality solutions than heuristics based on network optimization). This is illustrated by the example of Figure 15. Hypercube suppresses an extra cell (X), whereas

```
**Algorithm** *Shortest-paths Heuristic for CSP (Table,$\mathcal{P}$, $upl$, $lpl$)*
    $\mathcal{S} = \emptyset$;  $Clpl_i = 0$,  $Cupl_i = 0$,  $i \in \mathcal{P}$;
    **for_each** $p \in \mathcal{P}$ **do**
        Find source and target nodes of primary arc $x_p^+ = (s, t)$;
        **for_each** type of protection level $* \in \{lpl, upl\}$ **do**
                $\mathcal{TT} = \emptyset$;
                **while** $(C*_p < *_p)$ **do**
                    Set arc costs;
                    Compute the shortest path $SP$ from $t$ to $s$;
                    **if** $SP$ is empty **then**
                        // *check whether CSP instance is infeasible*
                    **end_if**
                    $\mathcal{T} = \{$cells associated with arcs $\in SP\}$;
                    $\mathcal{S} := \mathcal{S} \cup \mathcal{T} \setminus \mathcal{P}$;
                    Compute $\gamma^+$ and $\gamma^-$;
                    Update $Clpl_i$ and $Cupl_i$, $i \in (\mathcal{P} \cap \mathcal{T}) \cup \{p\}$;
                    $\mathcal{TT} := \mathcal{TT} \cup \mathcal{T}$;
                **end_while**
        **end_for_each**
    **end_for_each**
    **Return:** $\mathcal{S}$;
**End_algorithm**
```

Figure 14: Outline of shortest paths heuristic for CSP.

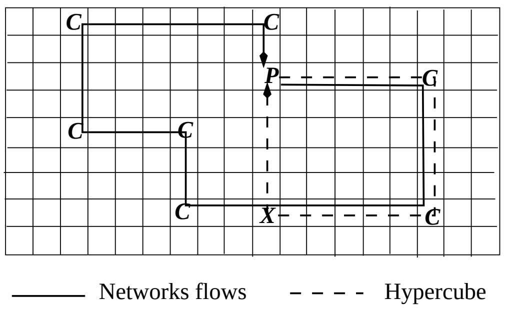

Figure 15: Illustration of the hypercube method on a two-dimensional table. P is the primary cell; C are previously removed complementary or secondary cells; X is an extra cell removed by hypercube.

a network flows heuristic would notice that primary cell P is already protected by a set of previously suppressed complementary cells C. Although it is efficient, the hypercube method in practice tends to over-suppress cells and, moreover, it does not guarantee a feasible solution (indeed, it may provide a solution with underprotected cells). Some of the above drawbacks are also shared by the other heuristic, named *Hitas* (deWolf, 2002). That approach decomposes a general k-dimensional hierarchical table in a tree of $(k-l)$-dimensional non-hierarchical subtables, $l = 0, \dots, k-1$, and locally protects them by the approach of Subsection 4.1. The practical implementation of Hitas can only deal with (hierarchical) tables with $k \leq 3$, since the approach of Subsection 4.1 is not efficient for (real-world instances of) higher dimensional tables. Since some linking constraints between subtables are removed, the final solution is not guaranteed to be feasible. However, the quality of the solutions is in general acceptable, and the overall procedure is quite fast. It is worth noting that, in spite of their infeasibility issues, both hypercube and Hitas provide solutions that avoid the singleton problem, making them, in practice, good choices for data owners.

4.4 Computational evaluation of approaches for CSP

It is not easy to compare the approaches of previous subsections 4.1–4.3, since codes are not freely available. However, most of them have been included in the τ-Argus software (Hundepool et al., 2008), freely available from `http://neon.vb.cbs.nl/casc/tau.htm`, and being used by several European NSAs. In particular, this package includes the Benders decomposition of Fischetti and Salazar-González (2001), the shortest paths heuristic of Castro (2007a), and the two (infeasible) heuristics of deWolf (2002) and Giessing and Repsilber (2002). These four approaches can be tested with the same set of tables, generated within the τ-Argus common interface.

Table 1 shows the dimensions and results obtained on a set of six 1H2D private real-world instances (Castro, 2007a). The information for instances CBS* and DES* was reported, respectively, by Statistics Netherlands and Statistics Germany. They were obtained with the τ-Argus package, running the four available solution approaches, and using Xpress as the MILP solver for Benders and Hitas. The information provided for each instance is the total number of cells n, the number of primary cells $|\mathcal{P}|$, and the CPU time (columns "CPU") and the best objective function value achieved (columns f^*). Problems CBS* and DES* were solved, respectively, on a 1.5 GHz Pentium-4 and a 900MHz Pentium-3 processor. From Table 1, it can be concluded that, although they do not guarantee optimal solutions, heuristics are much faster than Benders decomposition for "complex" tables (i.e., non-two-dimensional tables, in particular 1H2D tables), and in some cases they provide similar (even better, due to the time limit of one hour set for these executions) objective functions. It is worth to note that solutions provided by Hitas and hypercube are not guaranteed to be feasible, while Benders and the shortest paths heuristics always provide a feasible solution.

Since results of Table 1 are not reproducible, because instances are private, an additional set of two public tables has been tested. The information for these two tables is reported in Table 2, and the meaning of the columns is the same than for Table 1. These two 1H2D magnitude tables are obtained from the example microdata file distributed with τ-Argus, by crossing variables "IndustryCode"×"Size", and reporting information for either "Var1" or "Var2" as output variables. CPLEX was used as the MILP solver for Benders and Hitas

Table 1: Results for private real-world tables from Statistics Netherlands and Statistics Germany.

			Benders		shortest paths		Hitas		hypercube	
Instance	n	$\lvert\mathcal{P}\rvert$	f^*	CPU	f^*	CPU	f^*	CPU	f^*	CPU
CBS1	6399	570	4.85e+6	>3600	4.84e+6	4	5.85e+6	12	11.8e+6	6
CBS2	172965	68964	(1)		2.96e+10	403	1.31e+10	1151	24.9e+10	177
DES1	460	18	0.90e+6	93	0.87e+6	6	1.68e+6	1	43.2e+6	2
DES2	1050	61	2.41e+7	98	2.44e+7	4	2.57e+7	4	4.06e+7	4
DES3	8230	994	10.2e+7	618	12.9e+7	10	9.41e+7	35	42.2e+7	9
DES4	16530	2083	(2)		1.83e+8	21	1.54e+8	38	5.98e+8	14
DES4a	29754	3494	(2)		11.9e+7	65	5.95e+7	119	33.8e+7	24

(1) not tried; (2) failed

Table 2: Results for magnitude 1H2D public tables from microdata file of τ-Argus distribution, obtained crossing variables "IndustryCode"×"Size", and reporting information for either "Var1" or "Var2".

			Benders		shortest paths		Hitas		hypercube	
Instance	n	$\lvert\mathcal{P}\rvert$	f^*	CPU	f^*	CPU	f^*	CPU	f^*	CPU
Var1	6399	657	18.7e+6	604(1)	20.4e+6	1	25.7e+6	4	54.2e+6	4
Var2	6399	1018	6.85e+6	605(1)	8.26e+6	2	8.75e+6	6	14.3e+6	3

(1) stopped after default τ-Argus time limit of 10 minutes reached

approaches. All the runs have been performed on a Linux Dell Precision T5400 workstation with 16GB of memory and four Intel Xeon E5440 2.83 GHz processors, without exploitation of parallelism capabilities. As it is observed, Benders requires much more CPU time than the other heuristic approaches, but as expected it provides a better objective. The default time limit of 10 minutes of CPU (which is the minimum allowed limit by the software) was reached by Benders. The shortest paths heuristic provides better results than the other heuristics with less CPU time, and it is guaranteed to provide a feasible solution. On the other hand, if the table was more complex (instead of 1H2D) the shortest paths heuristic could not be used. For end-users, the availability of several different approaches (which represent a tradeoff between solution time and solution quality) is very positive: they just have to select the appropriate one for their particular needs.

5 Controlled tabular adjustment

The purpose of *controlled tabular adjustment* (also known as *minimum-distance controlled tabular adjustment* or simply *CTA*) is to find the closest safe table to the original one (i.e., the closest table that meets some protection levels). Since CTA is a perturbative method, its goal is to publish a table where the values of sensitive cells have been modified, such that any attempt for retrieving confidential information from individual respondents will fail (i.e., it will provide wrong values, far enough from the real ones). This risk model, although different from the one used by CSP, achieves the same goal of preserving confidentiality, and it is thus equally valid. Although CTA is a recent approach, compared to CSP, it is gaining recognition; indeed in some occasions end users prefer a slightly perturbed value to an empty

Original table

	P_1	P_2	P_3	
M_1	20	24	28	72
M_2	38	38	**40**	116
M_3	40	39	42	121
	98	101	110	309

Adjusted table,
lower protection sense

	P_1	P_2	P_3	
M_1	15	24	33	72
M_2	43	38	**35**	116
M_3	40	39	42	121
	98	101	110	309

Adjusted table,
upper protection sense

	P_1	P_2	P_3	
M_1	25	24	23	72
M_2	33	38	**45**	116
M_3	40	39	42	121
	98	101	110	309

Figure 16: Original table with sensitive cell in boldface, of lower and upper protection levels equal to five. Protected tables with "lower protection sense" and "upper protection sense" (i.e., value of sensitive is respectively reduced and increased by five units).

cell (Zayatz, 2009). CTA is considered and discussed as a new emerging protection method in Hundepool et al. (2010). Figure 16 illustrates CTA on a small two-dimensional table with one sensitive cell in boldface, with lower and upper protection levels equal to five (left table of Figure 16). Depending on the "protection sense" of the sensitive cell, either "lower" or "upper", which has to be decided, the value to be published for this cell will be respectively less or equal than the original cell value minus the lower protection level, or greater or equal than the original cell value plus the upper protection level. In the example of Figure 16, if the protection sense is "lower", then the value published for the sensitive cell should be less or equal than 35; the optimal adjusted table for this case is shown in the middle table of Figure 16. If the protection sense is "upper", then the value must be greater or equal than 45, as shown in the right table of Figure 16.

CTA was introduced in the manuscript Dandekar and Cox (2002) and, independently and in an extended form, in Castro (2006) (in the latter it was named minimum-distance controlled perturbation method). The parameters that define any CTA instance are the same than for CSP, i.e.:

- A general table $a_i, i = 1, \ldots, n$, with m linear relations $Aa = b$.
- Upper and lower bounds u and l for the cell values, assumed to be known by any attacker: $l \leq a \leq u$.
- Vector of nonnegative weights associated to the cell perturbations $w_i,\ i = 1, \ldots, n$.
- Set $\mathcal{P} \subseteq \{1, \ldots, n\}$ of sensitive cells.
- Lower and upper protection levels for each primary cell lpl_p and upl_p $p \in \mathcal{P}$.

CTA finds the safe table x closest to a, using some distance $L_{(w)}$:

$$\begin{array}{ll} \min_x & ||x - a||_{L(w)} \\ \text{s. to} & Ax = b \\ & l \leq x \leq u \\ & x_p \leq a_p - lpl_p \text{ or } x_p \geq a_p + upl_p \quad p \in \mathcal{P}. \end{array} \tag{16}$$

Defining $z = x - a$, $l_z = l - a$ and $u_z = u - a$, (16) can be recast in terms of deviations:

$$\begin{array}{ll} \min_z & ||z||_{L(w)} \\ \text{s. to} & Az = 0 \\ & l_z \leq z \leq u_z \\ & z_p \leq -lpl_p \text{ or } z_p \geq upl_p \quad p \in \mathcal{P}. \end{array} \tag{17}$$

To model the "or" constraints it is necessary to consider binary variables $y_p \in \{0,1\}$, $p \in \mathcal{P}$, such that $y_p = 1$ if cell p is "upper protected" (i.e, $z_p \geq upl_p$), and $y_p = 0$ if it is "lower protected" ($z_p \leq -lpl_p$). For the particular case of distance L_1, it is also needed a pair of variables z_i^+ and z_i^-, such that $z_i = z_i^+ - z_i^-$ and $|z_i| = z_i^+ + z_i^-$. The resulting MILP model is

$$\begin{array}{ll}
\min\limits_{z^+,z^-} & \sum\limits_{i=1}^{n} w_i(z_i^+ + z_i^-) \\
\text{s. to} & A(z^+ - z^-) = 0 \\
& 0 \leq z_i^+ \leq u_{z_i} \quad i \notin \mathcal{P} \\
& 0 \leq z_i^- \leq -l_{z_i} \quad i \notin \mathcal{P} \\
& upl_i y_i \leq z_i^+ \leq u_{z_i} y_i \quad i \in \mathcal{P} \\
& lpl_i(1-y_i) \leq z_i^- \leq -l_{z_i}(1-y_i) \quad i \in \mathcal{P} \\
& y_i \in \{0,1\} \quad i \in \mathcal{P}.
\end{array} \tag{18}$$

Problem (18) has $|\mathcal{P}|$ binary variables, $2n$ continuous variables and $m+4|\mathcal{P}|$ constraints. The size of (18) is much less than that of the the cell suppression formulation (7). For instance, for a table of 8000 cells, 800 primaries, and 4000 linear relations, CTA formulates a MILP of 800 binary variables, 16000 continuous variables and 7200 constraints (these figures would be 8000, 12,800,000 and 32,000,000 for CSP).

The benefits of CTA are not limited to a smaller size of the resulting MILP problem. CTA can be easily extended with constraints to meet some data quality criteria (Cox et al., 2005). It has also been experimentally observed that the information loss of CTA solutions is comparable (in some instances even better) than that of CSP. For instance, in Castro and Giessing (2006), it was observed that for a real-world instance from Statistics Germany, the number of cells with large deviations provided by CTA was six, whereas the number of suppressed cells by CSP was 20. If we consider that a cell with a "large" deviation in CTA is equivalent to a cell whose value is "suppressed" in CSP (since the information provided by this cell is "small"), we may conclude that the information loss of the CTA solution was lower in that situation.

Being a recent method, there is not too much literature about solution approaches for CTA. We outline in next subsections some of the attempts currently being performed.

5.1 Use of state-of-the-art solvers for CTA

Because of the relatively small size of the CTA formulation (18), it is possible to apply (usually not straightforwardly, but tuning some parameters) state-of-the-art MILP solvers. Such an implementation was developed using both CPLEX and Xpress in a package to be used and funded by Eurostat (Castro et al., 2009). Table 3 shows the results obtained with the commercial state-of-the-art solver Xpress on a set of four real-world instances provided by Eurostat, and processed by Statistics Germany and Statistics Netherlands. These instances can be considered difficult, since they have a complex structure. They are related to structural business statistics, for NACE sectors C, D and E. (The NACE code system is the European standard for industry classifications. Sectors C, D and E correspond respectively to "manufacturing", "electricity, gas, steam and air conditioning supply" and "water supply; sewerage; waste management and remediation activities".) Columns n, $|\mathcal{P}|$ and m provide

Table 3: Results with Xpress for some real data of structural business statistics from Eurostat.

Instance	n	$\|\mathcal{P}\|$	m	objective	gap(%)	CPU (sec)
sbs-E	1430	382	991	109130	2.9	4.3
sbs-C	4212	1135	2580	314950	2.0	57
sbs-D$_a$	28288	7142	13360	414474	4.9	11548
sbs-D$_b$	28288	7131	13360	407665	4.9	19510

the number of cells, sensitive cells and linear relations of the table. Columns objective, gap and CPU show the final value of the objective function (weighted perturbations for all the cells), optimality gap, and CPU time in seconds. The optimality gap is defined as

$$gap = \frac{best - lb}{10^{-10} + |best|} \cdot 100\%,$$

best being the best current solution, and *lb* the best current lower bound. All the runs were carried out on a Linux Dell PowerEdge 6950 server with four AMD Opteron 8222 3.0 GHZ processors without exploitation of parallelism capabilities (since parallelism was not exploited, a modern PC would provide faster executions than those reported in Table 3). The required optimality gap was of 5% for all the executions. We see that the smallest execution was solved in seconds, the medium-sized one in less than a minute, and the two largest required few hours of CPU. However, the solution of larger instances, with tighter optimality gaps (i.e., close to 0%) result in executions of more than one day of CPU, and specialized solution approaches are needed for them.

5.2 Benders reformulation of CTA

One specialized solution approach is, as for CSP, Benders reformulation (not decomposition in this case) for CTA, where the master problem is formulated in terms of the binary variables (protection senses) and the subproblem just finds a feasible pattern of adjustments for these protection senses. It can be shown (Castro and Baena, 2008) that, applying Benders method to (18), the formulation of the subproblem (for some $y_i, i \in \mathcal{P}$ values) is:

$$\begin{array}{ll} \max\limits_{\mu_u^+,\mu_u^-,\mu_l^+,\mu_l^-} & -{\mu_u^+}^T u^+ - {\mu_u^-}^T u^- + {\mu_l^+}^T l^+ + {\mu_l^-}^T l^- \\ \text{s. to} & \begin{pmatrix} A^T \\ -A^T \end{pmatrix} \lambda - \begin{pmatrix} \mu_u^+ \\ \mu_u^- \end{pmatrix} + \begin{pmatrix} \mu_l^+ \\ \mu_l^- \end{pmatrix} = \begin{pmatrix} w \\ w \end{pmatrix} \\ & \mu_u^+, \mu_u^-, \mu_l^+, \mu_l^- \geq 0, \end{array} \qquad (19)$$

where l^+, l^-, u^+, u^- provide the lower and upper bounds of z^+ and z^- once binary variables $y \in \mathbb{R}^p$ are fixed, $\mu_u^+, \mu_u^-, \mu_l^+, \mu_l^- \in \mathbb{R}^n$ are the vectors of Lagrange multipliers for these lower and upper bounds, and $\lambda \in \mathbb{R}^m$ is the vector of Lagrange multipliers for the table linear

Table 4: Results with Benders reformulation for CTA on two-dimensional tables.

	dimensions			Benders		CPLEX	
Instance	n	$\lvert\mathcal{P}\rvert$	m	CPU	Iter.	CPU	MIP iter.
random16	22801	10000	302	4.26	9	8.29	23228
random17	22801	18500	302	3.24	5	19.08	41301
random18	15251	13000	252	1.69	4	9.16	28959
random19	15251	11000	252	2.02	5	8.53	24856
random20	22801	18500	302	3.18	5	17.3	413013

relations. The formulation of the master problem is

$$\begin{array}{lll}
\min\limits_{\theta,y} & \theta & \\
\text{s. to} & \sum\limits_{h\notin\mathcal{P}}(-\mu^{+,i}_{u_h}u_{z_h}+\mu^{-,i}_{u_h}l_{z_h})+\sum\limits_{h\in\mathcal{P}}(\mu^{-,i}_{u_h}l_{z_h}+\mu^{-,i}_{l_h}lpl_h)+ & \\
& +\sum\limits_{h\in\mathcal{P}}(-\mu^{+,i}_{u_h}u_{z_h}-\mu^{-,i}_{u_h}l_{z_h}+\mu^{+,i}_{l_h}upl_h-\mu^{-,i}_{l_h}lpl_h)y_h\le\theta & i\in\mathcal{I} \\
& \sum\limits_{h\notin\mathcal{P}}(-v^{+,j}_{u_h}u_{z_h}+v^{-,j}_{u_h}l_{z_h})+\sum\limits_{h\in\mathcal{P}}(v^{-,j}_{u_h}l_{z_h}+v^{-,j}_{l_h}lpl_h)+ & \\
& +\sum\limits_{h\in\mathcal{P}}(-v^{+,j}_{u_h}u_{z_h}-v^{-,j}_{u_h}l_{z_h}+v^{+,j}_{l_h}upl_h-v^{-,j}_{l_h}lpl_h)y_h\le 0 & j\in\mathcal{J} \\
& y_h\in\{0,1\}\quad h\in\mathcal{P}, &
\end{array}\tag{20}$$

where θ is the computed lower bound of the optimal objective, and $\mathcal{I}$ and $\mathcal{J}$ are the sets of optimality and feasibility cuts (associated to vertices and rays of the polyhedron of the feasible region of (19)).

This approach was tested on a set of random two-dimensional instances (Castro and Baena, 2008) obtained with the generator in Castro (2007a). All runs were carried on a Sun Fire V20Z server with two AMD Opteron processors (without exploiting parallelism capabilities), and under the Linux operating system. Table 4 shows the instance dimensions (number of cells n, number of sensitive cells $|\mathcal{P}|$, number of table linear relations m), and the computational results (CPU times with both Benders and CPLEX branch-and-cut, number of Benders iterations "Iter.", and number of CPLEX simplex iterations "MIP iter."). CPLEX was used for the solutions of (19) and (20). From this table it can be shown that Benders reformulation is more efficient than a branch-and-cut approach for two-dimensional tables. However, this does not hold for more complex tables, as the number of Benders iterations grows significantly. This is the same behaviour observed for the Benders decomposition approach for CSP of Subsection 4.1: it could efficiently solve large two-dimensional tables (Fischetti and Salazar-González, 1999), but it was only applied to smaller three-dimensional and other complex tables (Fischetti and Salazar-González, 2001).

Other specialized (exact or heuristic) approaches are being considered for larger CTA instances. Some preliminary work has been started improving the standard Benders reformulation of (Castro and Baena, 2008), and on block coordinate-descent heuristics (González and Castro, 2011).

6 Conclusions

This paper presented some of the techniques for tabular data protection, focusing on CSP and CTA. It also outlined the main successful optimization approaches currently being applied. They are used in real-world by NSAs for the protection of released tables. All of those approaches share, at different degrees, the same computational drawbacks: they result in large difficult MILP optimization problems. Current research for improving the solution of these MILPs is being undertaken, mainly for the most recent method, CTA. There are other alternative protection methods, like *interval protection* or *partial cell suppression* which result in a very large, even massive, linear programming problem. However, being a continuous optimization problem, recent specialized interior-point methods for structured problems (Castro, 2007b; Castro and Cuesta, 2010) are expected to be a very efficient alternative. This is research to be conducted in the near future. Adding to current CSP methods the solutions outlined in Section 4 to deal with the singleton problem is also part of the work to be done in this challenging field.

Acknowledgments

This work has been supported by grants MTM2009-08747 of the Spanish Ministry of Science and Innovation, and SGR-2009-1122 of the Government of Catalonia. The author thanks two anonymous reviewers for helpful suggestions and comments.

References

R.K. Ahuja, T.L. Magnanti and J.B. Orlin (1993), *Network flows. Theory, Algorithms and Applications*, Prentice Hall, Upper Saddle River, NJ.

M. Bacharach (1966), Matrix rounding problems, *Management Science*, 9, 732-742.

J.F. Benders (2005), Partitioning procedures for solving mixed-variables programming problems, *Computational Management Science*, 2 3–19. English translation of the original paper appeared in *Numerische Mathematik*, 4 238–252 (1962).

F.D. Carvalho, N.P. Dellaert and M.D. Osório (1994), Statistical disclosure in two-dimensional tables: general tables, *Journal of the American Statistical Association*, 89 1547–1557.

J. Castro (2002), Network flows heuristics for complementary cell suppression: an empirical evaluation and extensions, *Lecture Notes in Computer Science*, 2316 59–73.

J. Castro (2004), A fast network flows heuristic for cell suppression in positive tables, *Lecture Notes in Computer Science*, 3050 136-148.

J. Castro (2005), Quadratic interior-point methods in statistical disclosure control, *Computational Management Science*, 2(2) 107-121.

J. Castro (2006), Minimum-distance controlled perturbation methods for large-scale tabular data protection, *European Journal of Operational Research*, 171 39-52

J. Castro (2007a), A shortest-paths heuristic for statistical data protection in positive tables, *INFORMS Journal on Computing*, 19(4) 520-533.

J. Castro (2007b), An interior-point approach for primal block-angular problems, *Computational Optimization and Applications*, 36 195–219.

J. Castro and D. Baena (2008). Using a mathematical programming modeling language for optimal CTA, *Lecture Notes in Computer Science*, 5262 1–12.

J. Castro and J. Cuesta (2010), Quadratic regularizations in an interior-point method for primal block-angular problems, *Mathematical Programming*, DOI:10.1007/s10107-010-0341-2, in press.

J. Castro and S. Giessing (2006), Testing variants of minimum distance controlled tabular adjustment, in *Monographs of Official Statistics. Work session on Statistical Data Confidentiality*, Eurostat-Office for Official Publications of the European Communities, Luxembourg, 2006, 333-343. ISBN 92-79-01108-1

J. Castro, A. González and D. Baena (2009), User's and programmer's manual of the RCTA package, Technical Report DR 2009/01, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, 2009.

L.H. Cox (1995), Network models for complementary cell suppression, *Journal of the American Statistical Association*, 90 1453–1462.

L.H. Cox and J.A. George (1989), Controlled rounding for tables with subtotals, *Annals of Operations Research*, 20 141–157.

L.H. Cox, J.P. Kelly and R. Patil (2005), Computational aspects of controlled tabular adjustment: algorithm and analysis. B. Golden, S. Raghavan, E. Wassil, eds. *The Next wave in Computer, Optimization and Decision Technologies*, Kluwer, Boston, MA, 45–59.

J. Daalmans, T. de Waal, An improved formulation of the disclosure auditing problem for secondary cell suppression, *Transactions on Data Privacy*, 3 217–251.

R.A. Dandekar and L.H. Cox (2002), Synthetic tabular data: An alternative to complementary cell suppression, manuscript, Energy Information Administration, US Department of. Energy.

N.P. Dellaert and W.A. Luijten (1999), Statistical disclosure in general three-dimensional tables, *Statistica Neerlandica*, 53 197–221.

P.P. de Wolf (2002), HiTaS: A heuristic approach to cell suppression in hierarchical tables, *Lecture Notes in Computer Science* 2316 74–82.

J. Domingo-Ferrer and L. Franconi (eds.) (2006), *Lecture Notes in Computer Science. Privacy in Statistical Databases* 4302, Springer, Berlin.

J. Domingo-Ferrer and J. M. Mateo-Sanz (2002), Practical data-oriented microaggregation for statistical disclosure contro, *IEEE Transactions on Knowledge and Data Engineering* 14, 189-201.

J. Domingo-Ferrer and E. Magkos (eds.) (2010), *Lecture Notes in Computer Science. Privacy in Statistical Databases* 6344, Springer, Berlin.

J. Domingo-Ferrer and Y. Saigin (eds.) (2008), *Lecture Notes in Computer Science. Privacy in Statistical Databases* 5262, Springer, Berlin.

J. Domingo-Ferrer and V. Torra (2002), A Critique of the Sensitivity Rules Usually Employed for Statistical Table Protection, *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 10(5), 545–556.

J. Domingo-Ferrer and V. Torra (eds.) (2004), *Lecture Notes in Computer Science. Privacy in Statistical Databases* 3050, Springer, Berlin.

M. Fischetti and J.J. Salazar-González (1999), Models and algorithms for the 2-dimensional cell suppression problem in statistical disclosure control. *Mathematical Programming*, 84 283–312.

M. Fischetti and J.J. Salazar-González (2001), Solving the cell suppression problem on tabular data with linear constraints, *Management Science* 47 1008–1026.

S. Giessing and D. Repsilber (2002), Tools and strategies to protect multiple tables with the GHQUAR cell suppression engine, *Lecture Notes in Computer Science* 2316 181–192.

A. González and J. Castro (2011), A heuristic block coordinate descent approach for controlled tabular adjustment, *Computers and Operations Research*, doi:10.1016/j.cor.2011.02.008, in press.

S.L. Hansen and S. Mukherjee (2003), A polynomial algorithm for optimal univariate microaggregation, *IEEE Transactions on Knowledge and Data Engineering* 15, 1043–1044.

A. Hundepool, A. van de Wetering, R. Ramaswamy, P.P. de Wolf, S. Giessing, M. Fischetti, J.J. Salazar-González, J. Castro and P. Lowthian (2008), *τ-Argus User's Manual*, Statistics Netherlands. Available on-line at `http://neon.vb.cbs.nl/casc/Software/TauManualV3.3.pdf`.

A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Naylor, E. Schulte-Nordholt, G. Seri and P.P. de Wolf (2010), *Handbook on Statistical Disclosure Control (v. 1.2)*, Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control. Available on-line at `http://neon.vb.cbs.nl/casc/SDC_Handbook.pdf`.

R. Jewett (1993), Disclosure analysis for the 1992 economic census, manuscript, Economic Programming Division, U.S. Bureau of the Census, Washington, DC.

J.P. Kelly, B.L. Golden and A.A. Assad (1992), Cell suppression: disclosure protection for sensitive tabular data. *Networks*, 22 28–55.

K. Muralidhar and R. Sarathy (2006), Data shuffling: a new masking approach for numerical data, *Management Science*, 52, 658–570.

D. Robertson (2000), Improving Statistics Canada's cell suppression software (CONFID), *Proceedings in Computational Statistics* 2000 (eds. J.G. Bethlehem and P.G.M. Van der Heijden), Physica-Verlag, New York, 403–408.

D.A. Robertson and R. Ethier (2002), Cell suppression: experience and theory, *Lecture Notes in Computer Science*, 2316 8–20.

J.J. Salazar-González (2004), Mathematical models for applying cell suppression methodology in statistical data protection, *European Journal of Operational Research* 154 740–754.

J.J. Salazar-González (2006), Controlled rounding and cell perturbation: statistical disclosure limitation methods for tabular data, *Mathematical Programming* 105 583–603.

J.J. Salazar-González (2008), Statistical confidentiality: Optimization techniques to protect tables, *Computers and Operations Research* 35 1638–1651.

L. Willenborg and T. de Waal (eds.) (2000) *Lecture Notes in Statistics. Elements of Statistical Disclosure Control* 155, Springer, New York.

L. Zayatz (2009), U.S. Census Bureau. Oral communication at Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality, Bilbao (Basque Country, Spain).