WP. 25
ENGLISH ONLY

UNITED NATIONS STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION STATISTICAL OFFICE OF THE EUROPEAN COMMUNITIES (EUROSTAT)

Joint UNECE/Eurostat work session on statistical data confidentiality
(Bilbao, Spain, 2-4 December 2009)

Topic (iv): Tools and software improvements

A PACKAGE FOR L1 CONTROLLED TABULAR ADJUSTMENT

Invited Paper

Prepared by Jordi Castro and José A. González, Universitat Politècna de Catalunya, Spain

A package for L_1 controlled tabular adjustment[1].

Jordi Castro, José A. González

Department of Statistics and Operations Research, Universitat Politècnica de Catalunya Jordi Girona 1–3, 08034 Barcelona, Catalonia,

(jordi.castro@upc.edu, jose.a.gonzalez@upc.edu)

Abstract. Minimum distance controlled tabular adjustment (CTA) is a recent methodology for the protection of tabular data. Given a table to be protected, the purpose of CTA is to find the closest table that guarantees the confidentiality of the sensitive cells. This is achieved by adding slight adjustments to the remaining cells, or a subset of them provided by the user, while the others (usually, the total cells) preserve their original values. This more restrictive variant is commonly denoted as RCTA. CTA (or RCTA) requires the solution of an optimization problem, but unlike other approaches, its dimension and complexity makes it tractable for state-of-the-art optimization solvers. Recently, the authors developed an implementation of CTA under a framework program funded by Eurostat, to be used for the protection of structural business statistics, and, in the near future, for the protection of animal production statistics. In this paper we discuss some of the implementation details of this CTA package, and provide results in the solution of some real-world instances.

1 Introduction

Controlled tabular adjustment (CTA) and other minimum distance related variants were suggested in Dandekar and Cox (2002) and Castro (2006) as a replacement to previous more complex approaches for tabular data protection. CTA can be seen as a method for generating a safe synthetic table, which is as close as possible to the original table. This is obtained by solving the following optimization problem: given a non-safe table, with a set of sensitive cells to be protected, find the closest safe table to the original one by adding the minimum amount of perturbations. Some of the good properties of CTA are:

- It can be applied to any table or set of linked tables. Even for complex and large tables a solution can be obtained in reasonable time.

[1]This work has been supported by grant MTM2006-05550 of the Spanish Ministry of Science and Education, and by Eurostat framework Eurostat framework contract 22100.2006.002-226.532.

Problem	constraints	continuous	binary
CSP/CRP	$2(m+2n)s$	$2ns$	n
CTA	$m+4s$	$2n$	s

(a)

Problem	constraints	continuous	binary
CSP/CRP	21,000,000	8,000,000	4,000
CTA	6,500	8,000	1,000

(b)

Table 1: (a) Sizes of optimization problems associated to cell suppression (CSP), controlled rounding (CRP) and CTA. (b) Figures for a particular table of 4000 cells, 1000 sensitive cells, and 2500 linear relations.

- From a computational point of view, the size of the resulting optimization problem is by far lower than for other well-known protection methods, such as the cell suppression problem (CSP) and the controlled rounding problem (CRP). For a table of n cells, s of them being sensitive, and m table linear relations, Table 1(a) shows the dimensions of the optimization problem formulated by CSP, CRP and CTA (number of constraints, and number of continuous and binary variables). For example, the particular figures for a table of 4000 cells, 1000 sensitive cells, and 2500 linear relations are provided in Table 1(b), clearly showing the different order of magnitude between the optimization problems.

- State-of-the-art solvers, such as CPLEX or XPRESS, can be applied to the solution of CTA. Other approaches like CSP or CRP require specialized solution methods, either optimal or heuristic.

- CTA is a perturbative method, like CRP. However, unlike CRP, CTA allows the user to preserve the value of some "significant" cells (i.e., cells with totals). Those cells are published in the protected table with the original values. For the remaining cells, it is even possible to fix the maximum allowed perturbation.

- CTA can be applied to both L_1 and Euclidean L_2 distances. L_2 distances provide mixed integer quadratic problems, which are more difficult to be solved, but reduce the largest deviations.

- CTA with L_1 instances does not guarantee integrality of the perturbations (i.e., they can be fractional values). Indeed, it is possible to obtain tables where the perturbations are fractional (e.g., three three-dimensional tables are modeled as a multicommodity flow problem, which is known not to provide integral flows). However, in most tables tested with the L_1 distance, the solution provided was integer without imposing integrality of perturbations (however, we do not claim the matrices were totally unimodular, which is sufficient for guaranteeing integrality). Even if perturbations were not integer, they would still be valid for magnitude tables.

- CTA allows minimum changes, even no changes, to published cells. This is a main difference with respect to CRP, which forces changes to some multiple of the predefined base number for all the cells. The quality of the solution provided by CTA is thus higher.

- Previous empirical testing (Castro and Giessing, 2006) showed the quality of the solution (measured as number of cells with large significant deviations) provided by CTA was comparable, even higher, than that obtained with CSP. Other quality criteria (Cox et al., 2004) can also be easily added to the CTA formulation.

CTA has been recently applied within a wider scheme for protection data of structural business status for Eurostat (project coordinated by Statistics Netherlands, with the participation of Destatis and Universitat Politècnica de Catalunya) (Giessing et al., 2009). Some of the main features of the package developed are presented in this paper. Computational results with real data initially provided by Eurostat, and later processed by Statistics Netherlands and Destatis, are also reported. For all the instances a state-of-the-art general mixed integer linear optimization (MILP) solver, such as CPLEX or XPRESS, was able to provide an optimal, or quasi-optimal solution, within a reasonable time limit. Specialized, hopefully more efficient procedures, for CTA are beyond the scope of this work. Some preliminary work has already been started by the authors (Castro and Baena, 2008; Castro and González, 2009).

The structure of the paper is as follows. Section 2 sketches the CTA formulation. Section 3 describes the package implementing CTA. Section 4 shows the computational results obtained in the solution of a set of real instances. Finally, section 5 lists some features to be developed in a newer version of the package.

2 Outline of minimum distance CTA

Any CTA instance, either with one table or a number of tables, can be represented by the following parameters:

- A set of cells $a_i, i = 1, \dots, n$, that satisfy some linear relations $Aa = b$ (a being the vector of a_i's), and a vector $w \in \mathbb{R}^n$ of positive weights for the deviations of cell values.

- A lower and upper bound for each cell $i = 1, \dots, n$, respectively l_{x_i} and u_{x_i}, which are considered to be known by any attacker. If no previous knowledge is assumed for cell i $l_{x_i} = 0$ ($l_{x_i} = -\infty$ if $a \geq 0$ is not required) and $u_{x_i} = +\infty$ can be used.

- A set $\mathcal{S} = \{i_1, i_2, \dots, i_s\} \subseteq \{1, \dots, n\}$ of indices of confidential cells.

- A lower and upper protection level for each confidential cell $i \in \mathcal{S}$, respectively lpl_i and upl_i, such that the released values satisfy either $x_i \geq a_i + upl_i$ or $x_i \leq a_i - lpl_i$.

CTA attempts to find the closest safe values $x_i, i = 1, \dots, n$, according to some distance L, that makes the released table safe. This involves the solution of the following optimization problem:

$$\begin{array}{ll} \min\limits_{x} & ||x - a||_L \\ \text{subject to} & Ax = b \\ & l_x \leq x \leq u_x \\ & x_i \leq a_i - lpl_i \text{ or } x_i \geq a_i + upl_i \quad i \in \mathcal{S}. \end{array} \tag{1}$$

Problem (1) can also be formulated in terms of deviations from the current cell values. Defining $z = x - a$, $l_z = l_x - a$, $u_z = u_x - a$, using the L_1 distance weighted by w, and introducing variables $z^+, z^- \in \mathbb{R}^n$ so that $z = z^+ - z^-$ and $|z| = z^+ + z^-$, the final MILP model for CTA is:

$$\min_{z^+, z^-, y} \quad \sum_{i=1}^{n} w_i (z_i^+ + z_i^-) \tag{2a}$$

$$\text{subject to} \quad A(z^+ - z^-) = 0 \tag{2b}$$

$$0 \leq z^+ \leq u_z, \quad 0 \leq z^- \leq -l_z \tag{2c}$$

$$y \in \{0, 1\}^s \tag{2d}$$

$$\left. \begin{array}{lll} upl_i \, y_i & \leq z_i^+ & \leq u_{z_i} y_i \\ lpl_i (1 - y_i) & \leq z_i^- & \leq -lz_i (1 - y_i) \end{array} \right\} i \in \mathcal{S} \tag{2e}$$

Constraints (2b) impose feasibility of the published perturbed table. Constraints (2c) guarantee perturbations are within allowed bounds. Constraints (2d)–(2e) force the new table is safe. When $y_i = 1$ the constraints mean $upl_i \leq z_i^+ \leq u_{z_i}$ and $z_i^- = 0$, thus the protection sense is "upper"; when $y_i = 0$ we get $z_i^+ = 0$ and $lpl_i \leq z_i^- \leq -l_{z_i}$, thus the protection sense is "lower".

3 Implementation of the CTA package

The package is provided both as an standalone application, and as a set of routines that can be called from the user's application (callable library). We here only refer to the standalone application. A description of the several routines in the callable library can be found in Castro et al. (2009).

The package reads instances in CSP format, already used in other methods implemented in the τ-Argus package (Hundepool, 2006). Once the input data file is created, the package can be called through:

```
main_CTA filename out_dir [-s s] [-g g] [-t t] [-p p] [-e e] [-b b] [-m m]
         [-v v] [-c c]
```

The first two parameters are mandatory, the remaining ones are optional, and may be entered in any order. Calling this main program with no parameters provides the following usage message:

```
usage: main_CTA filename out_dir [-s s] [-g g] [-t t] [-p p] [-e e] [-b b]
                 [-m m] [-v v] [-c c]
where    filename: instance file in csp format
         outdir: directory for output files (must exist!)
         s: solver  s= 'c' (CPLEX) or 'x' (XPRESS) (default 'x')
         f: stop at first feasible solution (y='n' (no) or 'y' (yes) (default 'n')
         g: % optimality gap (default g= 5%)')
         t: initial limit time in seconds for optimization (default t= 86400)
         p: preprocess sensitive cells   p='n' (no) or 'y' (yes) (default 'n')
         e: feasibility tolerance (e >= 1.0e-9, default e=1.0e-6)
         i: integrality tolerance (1>=i>=0, default is i= -1: solver default;
            i>=e in XPRESS)
         b: big value to be used, at most, for bounds on deviations
            (default b=Infinity; b= -1: automatically set by the code; if
            problems, set a decent big value as 1.0e+8)
         h: emphasis for XPRESS (h=-1,0,1,2,3, default is -1; quality 0--speed 3)
         m: mipemphasis for CPLEX (m=0,1,2,3,4, default is 0= balanced)
         v: variable selection criteria in CPLEX (v=-1,0,1,2,3,4, default is 0)
         c: check input table and solution c= 'n' (no) or 'y' (yes) (default 'y')
```

The meaning of some of the above options is as follows:

f: If yes, the package will stop once the first feasible solution has been found, and it will ask for more CPU time (if 0 is entered, it will definitely stop).

t: CPU time limit in seconds. The optimization will be stopped once this limit has been reached, and the package will ask for more CPU time (if 0 is entered, it will definitely stop).

g: Optimality gap measures the quality of the solution as a relative distance from the current solution to a known lower bound of the optimal solution. Setting g=0% asks for the real optimal solution, but it may be computationally very expensive. Increasing g (e.g., from the default 5% to 50%), the code will likely obtain a feasible sub-optimal solution quickly.

3	336	309	484	397	1529
25	3	$\mathbf{393}^{\mathbf{30}}_{\mathbf{40}}$	48	15	484
1	2	$\mathbf{137}^{\mathbf{14}}_{\mathbf{14}}$	145	107	392
55	$\mathbf{291}^{\mathbf{30}}_{\mathbf{15}}$	91	166	$\mathbf{212}^{\mathbf{21}}_{\mathbf{21}}$	815
84	632	930	843	731	3220

(a)

3	351	309	490	376	1529
25	3	423	18	15	484
1	2	113	169	107	392
55	276	85	166	233	815
84	632	930	843	731	3220

(b)

Figure 1: (a) Original table, with sensitive cells in boldface, and lower and upper protection levels as subscripts and superscripts; (b) Adjusted table after CTA

e: Feasibility tolerance, i.e., the degree in constraints/bounds violations allowed by the optimization procedure. See Subsection 3.1 for details.

i: Integrality tolerance, i.e., the amount by which the binary variables in the RCTA model can be different from 0 or 1, and still be considered 0 or 1. See Subsection 3.1 for details about this parameter.

c: If this parameter is "y" some simple checks about the input table and the solution obtained is performed and reported on the screen. These checks include feasibility of linear table relations, protection of sensible cells, lower and upper bounds of adjusted table values, and quality of internal optimization model variables (i.e., that no both the positive and negative variables z_i^+ and z_i^- of cell i are positive in the solution of the mathematical programming model (2)).

After solving the problem, the package returns three types of output: (1) output on screen, with minimum information about the instance features, and checks about the input and protected tables; (2) a file with the output of the optimization process provided by the solver; (3) a file with the safe table obtained.

For instance, Figure 1(a) shows a small table with sensitive cells in boldface, and lower and upper protection levels as, respectively, subscripts and superscripts. Coding this table in a file named, e.g., `example.in` and running

```
main_CTA {path_of_instance}/example.in {path_of_output_directory}
```

the safe table shown in Figure 1(b) is obtained. The output on screen for this execution is:

```
CTA instance:                    example
Number of cells:                 30
Number of sensitive cells:       4
Number of constraints:           11
Solver:                          XPRESS
```

```
XPRESS MIP emphasis:            -1
MIP optimality gap:             0.05
MIP time limit (seconds):       86400
Stop at first feasible:         n
Feasibility tolerance:          1e-06
Integrality tolerance:          solver default
Big-M:                          1e+120

Checking table relations for ORIGINAL values.
0 constraints not satisfied within provided tolerance.

At optimum:  Objective F.: 1.3656  Lower bound: 1.3656  Optimality gap: 0%

Checking table relations for CTA values.
0 constraints not satisfied within provided tolerance.
Checking cell protections.
0 unprotected sensitive cells in CTA solution.
Checking cell bounds.
0 violated cell bounds in CTA solution.
Checking cell perturbations.
0 wrong perturbations in CTA solution.
Optimal CTA table found (optimal within tolerances)
Total CPU time: 0.01
```

3.1 Guidelines for difficult CTA instances

As shown above, several package options allow the user to control the solution of the mathematical programming model of CTA. Unfortunately, no set of default options is in general valid for every CTA instance. This applies to both solvers, CPLEX and XPRESS. If difficulties appear in the solution of some instance, the main parameters to be adjusted are the following:

- **Feasibility tolerance.** This is the degree in constraints/bounds violations allowed by the optimization procedure. In CPLEX it must be greater or equal than 1.0e−9; in XPRESS it must be greater or equal than 0. If it is too tight (e.g., 1.0e−9) the solver may falsely conclude the problem is infeasible. By default 1.0e−6 is used. If the problem is reported as infeasible, and it is believed to be feasible, then the feasibility tolerance should be increased (e.g., to 1.0e−5, or 5.0e−5). However, this may affect the quality of the solution: the solver may report as optimal a solution that underprotects some cells. The explanation is the following: constraints (2e) impose

$$z_i^+ \le u_{z_i}\, y_i, \qquad z_i^- \le -l_{z_i}(1-y_i),$$

where u_{z_i} and $-l_{z_i}$ are the maximum cell deviations upwards and downwards, respectively. If the cell bounds are large, u_{z_i} and $-l_{z_i}$ may be large as well.

The above constraints force that, when $y_i = 1$ (protection sense is “upper”), the downwards deviation must satisfy $z_i^- \le -l_{z_i}(1 - y_i) = 0$. However, in practice, because of the feasibility tolerance, we may have $y_i = 1 - \epsilon$, and thus if $-l_{z_i} = M$, and M is a big-value, the constraint imposes $z_i^- \le -l_{z_i}(1 - y_i) = M(1 - (1 - \epsilon)) = M\epsilon > 0$. Therefore, we allow a downwards deviation in a cell that was “upper” protected, leading to an underprotection. A similar reasoning applies for “lower” protected cells (i.e., $y_i = \epsilon$ instead of $y_i = 0$).

Decreasing the feasibility tolerance, we reduce the above ϵ value, but we make the problem much harder, and the solver may report it as infeasible. A best option, if possible, would be to avoid big-values M for cell deviations, but this means the real cell bounds (lower and upper bounds) should be small. If they were about 1.0e+4 or 1.0e+5, the above underprotection issue would not appear. However, in practice, real tables contain very big cell values, and the above “small” bounds could not be possible. Whenever possible, the user should try to tight them, if she/he has information about the data. Setting an arbitrarily large upper bound is a bad practice. The package includes an option to automatically set a maximum bound for all deviations.

- **Integrality tolerance**. This is the amount by which the binary variables in the RCTA model can be different from 0 or 1, and still be considered 0 or 1. The CPLEX default is 1.0e−5; the XPRESS default is 5.0e−6. In CPLEX it must be a value greater or equal than 0; in XPRESS it must be greater or equal than the feasibility tolerance. This parameter is related with the above feasibility tolerance. Indeed combining both of them we may try to obtain feasible/optimal solutions with no underprotected cells. We discussed in previous item how to avoid underprotections by tuning the feasibility tolerance. The integrality tolerance provides a new possibility: if it is set to a very small value, e.g., 1.0e−10, we are asking for binary solutions that are far from 0 or 1 at most 1.0e−10. Therefore the problem with constraints $z_i^+ \le u_{z_i}\ y_i$ and $z_i^- \le -l_{z_i}(1 - y_i)$, explained above, may be avoided. Unfortunately, there are two drawbacks of this approach. First, it may significantly increase the solution time of the branch-and-cut procedure (very significantly, indeed). Second, XPRESS (unlike CPLEX) requires and integrality tolerance greater or equal than the feasibility tolerance. Then if we reduce the integrality tolerance, we must reduce the feasibility tolerance as well, and then the algorithm may falsely conclude the problem is infeasible.

4 Some computational results

The CTA package has been applied to several instances. In this work we only focus on four particular real-world datasets provided by Eurostat. These instances can be considered difficult, since they have a complex structure. They are related to struc-

Problem	n	s	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

Table 2: Results with XPRESS for some real data of structural business statistics (provided by Eurostat, and processed by Statistics Netherlands and Destatis).

tural business statistics, for different NACE sector (C, D and E). The dimensions of these instances, and the results obtained with the CTA package, are reported in Table 2. Columns n, s and m provide the number of cells, sensitive cells and linear relations of the table. Columns objective, gap and CPU show the final value of the objective functions (weighted perturbations for all the cells), optimality gap, and CPU time in seconds obtained with XPRESS. The optimality gap is defined as

$$gap = \frac{best - lb}{1 + |best|} \cdot 100\%,$$

best being the best current solution, and *lb* the best current lower bound. All the runs were carried 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 2). 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. One and seven cells remained underprotected for, respectively, instances sbs-D$_a$ and sbs-D$_b$, due to the effect described in Subsection 3.1. However, those underprotections were not significant, being in most cases of one unit (in cells with values of thousands). It is worth noting that CPLEX was not able to solve the two largest instances even tuning some of the parameters. For XPRESS, a feasibility tolerance of 10^{-6} and a maximum deviation of 10^{8} were set. Other combinations lead to wrong solutions (i.e., too many underprotected cells). In general, tuning the parameters is not necessary, but it could be for large complex instances.

5 Conclusions and future work

The package for CTA (RCTA) is a robust and flexible tool which relies on two state-of-the-art optimization solvers. For most medium-sized instances, it may provide a solution in reasonable time. The package is currently being extended with three new features: (1) ability to work with nonadditive tables, i.e., the resulting deviations will both protect the data and will make the published table additive; (2) a new

CTA model which allows “negative” protection limits, to be used with correlated sets of tables; (3) a tool for pseudo-automatic analysis of infeasible instances (i.e., a help for the user to know why an instance is reported as infeasible).

References

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

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

Castro, J., and Giessing, S., (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, 333–343.

Castro, J., and González, J.A. (2009). Block coordinate descent decomposition for statistical data protection using controlled tabular adjustment, Research report DR 2009-10, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya. Submitted.

Castro, J., González, J.A., and Baena, D. (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.

Cox, L. H., Kelly, J. P., and Patil, R. (2004). Balancing quality and confidentiality for multivariate tabular data, *Lecture Notes in Computer Science*, 3050, 87–98.

Dandekar, R.A., and Cox, L.H. (2002). Synthetic tabular data: an alternative to complementary cell suppression, manuscript, Energy Information Administration, U.S. Department of Energy. Available from the first author on request (`Ramesh.Dandekar@eia.doe.gov`).

Giessing, S., Hundepool, A., and Castro, J. (2009). Rounding methods for protecting EU-aggregates, in *Worksession on statistical data confidentiality. Eurostat methodologies and working papers*, Eurostat-Office for Official Publications of the European Communities, Luxembourg, 255–264.

Hundepool, A. (2006). The Argus software in CENEX, *Lecture Notes in Computer Science* 4302, 334–346.