A linear optimization based method for data privacy in statistical tabular data

Jordi Castro José A. González
Dept. of Stat. and Operations Research
Univ. Politècnica de Catalunya
Barcelona, Catalonia
jordi.castro@upc.edu jose.a.gonzalez@upc.edu

Research Report UPC-DEIO DR 2017-02
September 2016; updated February 2017, May 2017

Report available from http://www-eio.upc.es/~jcastro

A linear optimization based method for data privacy in statistical tabular data

Jordi Castro* José A. González
Dept. of Statistics and Operations Research
Universitat Politècnica de Catalunya
Jordi Girona 1–3, 08034 Barcelona
jordi.castro@upc.edu jose.a.gonzalez@upc.edu

National Statistical Agencies routinely disseminate large amounts of data. Prior to dissemination these data have to be protected to avoid releasing confidential information. Controlled tabular adjustment (CTA) is one of the available methods for this purpose. CTA formulates an optimization problem that looks for the safe table which is closest to the original one. The standard CTA approach results in a mixed integer linear optimization (MILO) problem, which is very challenging for current technology. In this work we present a much less costly variant of CTA that formulates a multiobjective linear optimization (LO) problem, where binary variables are pre-fixed, and the resulting continuous problem is solved by lexicographic optimization. Extensive computational results are reported using both commercial (CPLEX and XPRESS) and open source (Clp) solvers, with either simplex or interior-point methods, on a set of real instances. Most instances were successfully solved with the LO-CTA variant in less than one hour, while many of them are computationally very expensive with the MILO-CTA formulation. The interior-point method outperformed simplex in this particular application.

Keywords: linear optimization; interior-point methods; benchmarking; lexicographic optimization; data science; data privacy; statistical disclosure control;

AMS Subject Classification: 90C05, 90C29, 90C51, 90C90

1. Introduction

National Statistical Agencies (NSAs) release two different types of data: raw data, usually named microdata, and tabular data obtained by crossing two or more categorical variables of the microdata. NSAs face the tradeoff between publishing as much as possible information and at the same time to avoid that confidential information can be derived from data released. The field of *statistical disclosure control* comprises the set of available methods to reduce the disclosure risk. More details about this field can be found in the recent survey [6] and the monographs [20, 21].

Although tabular data report aggregated information—so individuals could be considered anonymized—there is a risk of disclosing confidential information. Tables of Figures 1 and 2 illustrate this situation with a simple case from [6]. Figure 1 shows a *frequency* table (i.e., a table providing the number of individuals within each cell) obtained by crossing variables "profession" and "municipality" of some microdata file. Figure 2 shows a *magnitude* table, i.e., one providing information about a third variable, overall salary (in 1000€) in this particular case. If cell (M_2, P_3) of Figure 1 was 1, then any person (including those who do not appear in the microdata file) would know the salary of this

*Corresponding author

	P_1	P_2	P_3	P_4	P_5	TOTAL
M_1	20	15	30	20	10	95
M_2	72	20	1 or 2	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.

individual is 22000€. This is named an *external attacker* scenario. If the value of this cell was 2, any of these two individuals could compute the salary of the other, becoming *internal attackers*. Even if the value of this cell was larger, e.g. 4, 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*. Indeed, the set of sensitive cells is a priori obtained by applying some sensitivity rules. Actually, the two more used rules are the (n, k) dominance rule (n individuals of a cell cannot contribute to more than a $k\%$ to the cell value) and the $p\%$ rule (the cell is considered sensitive if some individual can compute an estimate of the value of another individual within a $p\%$ precision). A detailed explanation of these rules is out of the scope of this work; additional details about them can be found in [20, 21].

If a table contains sensitive cells, NSAs have to apply some tabular data protection method prior to publication. In short, those methods, basically either suppress or perturb the table cell values. Formally, a tabular data protection method can be seen as a map F such that $F(T) = T'$, i.e., table T is transformed to another table T'. The two main requirements for F are: (1) T' should be "safe", and (2) the quality of T' should be high (or equivalently, the information loss should be small), i.e., analysis made with T' and T should be similar. The disclosure risk can be analyzed through the inverse map $T = F^{-1}(T')$: if not available or difficult to compute by any *data attacker*, then we may guarantee that F is safe [7].

As far as we know, the first publication describing a protection method for tabular data was [1]. Today, the most used method is likely *cell suppression* [5, 26], a *non-perturbative* method, where some cell values are removed from the table. Among the *perturbative* approaches we find *controlled tabular adjustment* (CTA) [4, 12], which is the focus of this work.

The goal of CTA—which will be formulated in Section 2—is, given a table with any structure, to find the closest *safe* table to the original one, according to some distance. This is achieved by adding to the original table a vector of deviations (or perturbations) of minimum norm that makes the released table safe. Safety is guaranteed by imposing that sensitive cells in the perturbed table are far enough from the original value. This means the cell value is either above *or* below some certain values, which requires a binary variable for the disjunctive constraint of each sensitive cell. The minimum amount of

above or below perturbations imposed to each sensitive cell are named, respectively, *upper protection* and *lower protection* levels. Changes in sensitive cells induce changes in the remaining ones to satisfy the value of marginal or total cells. This standard CTA method results in a difficult mixed integer linear optimization (MILO) problem.

CTA is one of the protection methods included and discussed in the recent monographs [20, 21]. Among the recent literature on CTA variants we find [9, 18]. Although CTA is not as widely used as cell suppression, it has been applied as one of the steps of other wider protection schemes, such as the pre-tabular protection method of [16]. In addition, some National Statistical Agencies are questioning current non-perturbative protection methods because "the task of balancing confidentiality and usability [...] is nearly impossible" [27]. Therefore there is a need for new methods, and this justifies the research on CTA and other approaches. Indeed, there is no actually any protection method that fits the needs of all NSAs in the world.

For real and large tables the MILO formulation of CTA results in a difficult optimization problem. It is worth noting that even the linear optimization (LO) problems obtained from large CTA instances by fixing the binary variables are very difficult for today state-of-the-art solvers. Indeed, some of these instances have been included in standard LO repositories [24]. Some heuristic approaches have been developed to compute feasible, good suboptimal solutions to the MILO-CTA problem with few computational resources. For instance a block coordinate-descent approach was developed in [17]. In [3] this approach was combined with a fix-and-relax heuristic. It should be stressed that "in practice, tabular data protection is the last stage of the "data cycle," and, in an attempt to meet publication deadlines, NSAs require methods that find fast solutions to protect large tables" [11]. For instance, recent online table generation services require real-time (i.e., a few seconds) protection procedures. Since optimization based procedures can not satisfy these low computational times for large tables (where large could mean millions of cells), some NSAs have devised specific methods relying on fast statistical procedures (in short, some noise is added to microdata, and tables are generated from these perturbed microdata)[25]. However those microdata-perturbing approaches ignore the linear constraints of the table, thus protected tables may present inconsistencies.

This work describes a new CTA variant, where the binary variables for lower or upper protection of each sensitive cell are a priori fixed. Rules for fixing the binary decisions will be presented in Section 5. The resulting continuous problem thus only focuses on minimizing the norm of the cell deviations vector. To guarantee the quality of the protected table, cell deviations are constrained to a tight percentage of the cell value, which, together with the fixed values of the binary variables, may lead to an infeasible solution. To guarantee feasibility we may change the right-hand-side of table relations, the bounds on deviations, or the protection levels. These three criteria, plus the minimization of the norm of the vector of deviations, lead to four different and opposite objectives, resulting in a four-objective optimization problem. This problem will be solved by lexicographic optimization. This is justified by two arguments. First, finding all the efficient solutions by a weighted sum scalarization may involve the solution of a large number of LO-CTA problems, which are known to be very challenging. For example, one of the two instances from the standard LO repository [24] that the recently released Google Glop LO solver could not solve [2] is actually a LO-CTA instance. Second, in this particular application there is a natural hierarchy on the importance of objectives, which justifies the use of lexicographic optimization. This multiobjective approach has been recently implemented within the FP7-INFRA-2010-262608 project funded by the European Union, with the participation, among others, of the NSAs of Germany, Netherlands, Finland, Sweden and Slovenia. The multiobjective LO-CTA software has been included in the τ-Argus package since version 4.1.0 [13, 19] (`http://neon.vb.cbs.nl/casc/tau.htm`), used by many

European national statistical institutes for the protection of tabular data.

The quality of the solution provided by the LO-CTA approach will in general be worse than that of the MILO-CTA formulation. However, it will give end users a chance to solve very large instances in moderate time. Other variants have been devised for the same purpose. For instance the CTA variant of [18] solves a sequence of LO problems by randomly assigning a value to some sensitive cells within its protection interval, and trying to adjust the rest of cells with minimum changes. Our multiobjective approach exhibits many differences with that one: (i) it solves only up to four LO problems; (ii) it allows changes in the bounds and constraints (not only on bounds); (iii) and most important, the values of sensitive cells usually guarantee the protection levels when this is the first criteria in the lexicographic order (this will be the case in all the computational experiments performed, since loose constraints were imposed for the other three criteria). If protection levels are not the first criteria in the lexicographic order, small variations in the protections may appear; but this can be acceptable, since protection intervals are anyway heuristically adjusted by data protectors. Alternative approaches, such as the one of [18], do not even pay attention to this fact, and just replace the sensitive cell by a value which is always within the protection interval, thus not safe.

The paper is organized as follows. Section 2 outlines the MILO-CTA model and it presents the LO-CTA problem without binary variables. Sections 3 and 4 introduce the multiobjective LO-CTA model, and how it was solved by lexicographic optimization. Section 5 describes the different procedures used to a priori fix the binary variables. Finally Section 6 provides extensive computational results in the solution of real-world tables using commercial and open source LO solvers; a comparison between LO-CTA and MILO-CTA is also reported, in terms of CPU time and quality of solutions.

2. Formulation of MILO-CTA and LO-CTA by fixing the binaries

Any CTA instance can be represented by the following parameters: *(i)* A set of cell values $a_i, i \in \mathcal{N} = \{1, \ldots, n\}$, that satisfy $\mathcal{M} = \{1, \ldots, m\}$ linear relations $Aa = b$, a being the vector of a_i's, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. (From now on, we will indistinctly refer to a_i as either "cell i" or "cell value i".) The particular structure of the table is defined by equations $Aa = b$. Each tabular constraint imposes that the inner cells have to be equal to the total or marginal cell, i.e., $\left(\sum_{i \in \mathcal{I}_j} a_i\right) - a_{t_j} = 0$, where $\mathcal{I}_j$ is the set of inner cells of relation $j \in \mathcal{M}$, and t_j is the index of the total cell of relation j. Any type of table can be modeled by these equations. *(ii)* A lower and upper bound for each cell $i \in \mathcal{N}$, respectively l_{a_i} and u_{a_i}, which are considered to be known by any attacker. If no previous knowledge is assumed for cell i, $l_{a_i} = 0$ ($l_{a_i} = -\infty$ if $a \geq 0$ is not required) and $u_{a_i} = +\infty$ can be used. The quality of the protected table can be forced by imposing tight cell bounds, although this may result in infeasibility issues. *(iii)* A set $\mathcal{S} = \{i_1, i_2, \ldots, i_s\} \subseteq \mathcal{N}$ of indices of confidential cells. *(iv)* Nonnegative lower and upper protection levels 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 $x_i \leq a_i - lpl_i$. *(v)* Nonnegative cell weights w_i, $i \in \mathcal{N}$, used in the definition of the objective function. These weights penalize perturbations from the original cell values in the released table.

CTA attempts to find the values $x_i, i \in \mathcal{N}$, which are closest to a_i according to some distance ℓ, that make the released table safe. The problem can be formulated in terms of

cell deviations by defining $z = x - a$, thus obtaining:

$$\begin{array}{ll} \min\limits_{z} & ||z||_\ell \\ \text{s. to} & Az = 0 \\ & l_a - a \leq z \leq u_a - a \\ & z_i \leq -lpl_i \text{ or } z_i \geq upl_i \quad i \in \mathcal{S}, \end{array} \tag{1}$$

$z \in \mathbb{R}^n$ being the vector of deviations.

Defining $z = z^+ - z^-$, $z^+ \in \mathbb{R}^n$ and $z^- \in \mathbb{R}^n$ being the vector of positive and negative deviations in absolute value, and introducing a vector of binary variables $y \in \mathbb{R}^s$ to model the disjunctive constraints (either "upper protection" $z_i \geq upl_i$ when $y_i = 1$ or "lower protection" $z_i \leq -lpl_i$ when $y_i = 0$), (1) can be written for the ℓ_1 distance as the following MILO problem:

$$\begin{array}{ll} \min\limits_{z^+,z^-,y} & \sum\limits_{i \in \mathcal{N}} w_i (z_i^+ + z_i^-) \\ \text{s. to} & A(z^+ - z^-) = 0 \\ & upl_i \; y_i \leq z_i^+ \leq (u_{a_i} - a_i) \; y_i \quad i \in \mathcal{S} \\ & lpl_i (1 - y_i) \leq z_i^- \leq (a_i - l_{a_i})(1 - y_i) \quad i \in \mathcal{S} \\ & 0 \leq z^+ \leq u_a - a \\ & 0 \leq z^- \leq a - l_a \\ & y_i \in \{0, 1\} \quad i \in \mathcal{S}. \end{array} \tag{2}$$

When $y_i = 1$ the constraints mean $upl_i \leq z_i^+ \leq (u_{a_i} - a_i)$ and $z_i^- = 0$, thus the protection direction is "upper"; when $y_i = 0$ we get $z_i^+ = 0$ and $lpl_i \leq z_i^- \leq (a_i - l_{a_i})$, thus protection direction is "lower". Model (2) is a difficult MILO problem for medium-large instances.

For large tables the MILO-CTA model is impractical if a quick solution is required. In those situations, an efficient alternative would be to a priori fix the binary variables, thus obtaining a continuous LO-CTA formulation. Possible infeasibilities in the resulting problem could be dealt with the approaches exposed in [10], some of them already used in the context of CTA [8]. Fixing in (2) the binary variables, the continuous ℓ_1-CTA approach can be formulated as the following LO problem:

$$\begin{array}{ll} \min\limits_{z^+,z^-} & \sum\limits_{i \in \mathcal{N}} w_i (z_i^+ + z_i^-) \\ \text{s. to} & A(z^+ - z^-) = 0 \\ & l^+ \leq z^+ \leq u^+ \\ & l^- \leq z^- \leq u^-, \end{array} \tag{3}$$

where $l^+, l^-, u^+, u^- \in \mathbb{R}^n$ are defined as

$$
\begin{aligned}
l_i^+ &= \begin{cases} upl_i & \text{if } i \in \mathcal{S} \text{ and } y_i = 1 \\ 0 & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 0) \end{cases} \\
u_i^+ &= \begin{cases} 0 & \text{if } i \in \mathcal{S} \text{ and } y_i = 0 \\ u_{a_i} - a_i & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 1) \end{cases} \\
l_i^- &= \begin{cases} lpl_i & \text{if } i \in \mathcal{S} \text{ and } y_i = 0 \\ 0 & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 1) \end{cases} \\
u_i^- &= \begin{cases} 0 & \text{if } i \in \mathcal{S} \text{ and } y_i = 1 \\ a_i - l_{a_i} & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 0). \end{cases}
\end{aligned}
\tag{4}
$$

3. Guaranteeing a good solution: multiobjective optimization problem

Fixing binary variables $y \in \{0,1\}^s$ in (2) the resulting continuous problem (3) may be infeasible. Infeasibilities can be avoided by modifying the feasible region of (3), i.e., changing the right hand side coefficients and/or the bounds of the variables. We can achieve feasibility by three means:

- Allowing changes in the table relations. Constraints $Az = 0$ are transformed to

$$Az + \alpha^+ - \alpha^- = 0, \quad \alpha^+ \in \mathbb{R}^m,\ \alpha^- \in \mathbb{R}^m,\ (\alpha^+, \alpha^-) \geq 0. \tag{5}$$

- Decreasing l_a and increasing u_a, whenever possible. Thus, instead of $l_a \leq a \leq u_a$ we will consider

$$l_a - \beta_l \leq a \leq u_a + \beta_u, \quad \beta_l \in \mathbb{R}^n,\ \beta_u \in \mathbb{R}^n,\ (\beta_l, \beta_u) \geq 0. \tag{6}$$

- Reducing upl_i and lpl_i, $i \in \mathcal{S}$. Thus, the new lower protection levels will be

$$lpl := lpl - \gamma_l, \quad upl := upl - \gamma_u, \quad \gamma_l \in \mathbb{R}^s,\ \gamma_u \in \mathbb{R}^s,\ (\gamma_l, \gamma_u) \geq 0. \tag{7}$$

Therefore, instead of the possibly infeasible (3), we can consider the always feasible problem

$$
\begin{array}{ll}
\min\limits_{\omega} & (f_1(z^+, z^-), f_2(\alpha^+, \alpha^-), f_3(\beta_l, \beta_u), f_4(\gamma_l, \gamma_u)) \\
\text{s. to} & A(z^+ - z^-) + \alpha^+ - \alpha^- = 0 \\
& l^+(\gamma_u) \leq z^+ \leq u^+(\beta_u) \\
& l^-(\gamma_l) \leq z^- \leq u^-(\beta_l) \\
& (\alpha^+, \alpha^-, \beta_l, \beta_u, \gamma_l, \gamma_u) \geq 0,
\end{array}
\tag{8}
$$

where

$$
\begin{aligned}
f_1(z^+, z^-) &= \sum_{i \in \mathcal{N}} w_i (z_i^+ + z_i^-) \\
f_2(\alpha^+, \alpha^-) &= e^\top (\alpha^+ + \alpha^-) \\
f_3(\beta_l, \beta_u) &= e^\top (\beta_l + \beta_u) \\
f_4(\gamma_l, \gamma_u) &= e^\top (\gamma_l + \gamma_u),
\end{aligned}
\tag{9}
$$

e is a vector of ones of appropriate length, $\omega = (z^+, z^-, \alpha^+, \alpha^-, \beta_l, \beta_u, \gamma_l, \gamma_u)$ denotes the full set of variables of the optimization problem, and

$$
\begin{aligned}
l_i^+(\gamma_{u_i}) &= \begin{cases} upl_i - \gamma_{u_i} & \text{if } i \in \mathcal{S} \text{ and } y_i = 1 \\ 0 & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 0) \end{cases} \\
u_i^+(\beta_{u_i}) &= \begin{cases} 0 & \text{if } i \in \mathcal{S} \text{ and } y_i = 0 \\ u_{a_i} + \beta_{u_i} - a_i & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 1) \end{cases} \\
l_i^-(\gamma_{l_i}) &= \begin{cases} lpl_i - \gamma_{l_i} & \text{if } i \in \mathcal{S} \text{ and } y_i = 0 \\ 0 & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 1) \end{cases} \\
u_i^-(\beta_{l_i}) &= \begin{cases} 0 & \text{if } i \in \mathcal{S} \text{ and } y_i = 1 \\ a_i - (l_{a_i} - \beta_{l_i}) & \text{if } (i \in \mathcal{N} \setminus \mathcal{S}) \text{ or } (i \in \mathcal{S} \text{ and } y_i = 0). \end{cases}
\end{aligned}
\tag{10}
$$

Abusing of notation, from now on we will use indistinctly $f_1(z^+, z^-)$ and $f_1(\omega)$, and similarly for f_2, f_3 and f_4. The feasible region of (8) will be written in compact form below as $\omega \in \Omega$.

Problem (8) is a multiobjective optimization problem. Its purpose is to obtain a protected table that minimizes f_1, but at the same time is feasible for the original constraints, i.e., it minimizes f_2, f_3 and f_4. Note that (f_1, f_2, f_3, f_4) are conflicting objectives: increasing one we can reduce the others. For instance, in the limit, increasing enough γ_l and γ_u (i.e., $\gamma_l = lpl$ and $\gamma_u = upl$, thus removing the protection levels) makes $z = \beta_l = \beta_u = 0$ and $\alpha^+ = \alpha^- = 0$. We also note that upper bounds for γ_{l_i} and γ_{u_i} are not needed, although this could result in negative protection levels upl_i and lpl_i. Indeed this would only happen if z_i^+ and z_i^- were negative, which would unnecessarily increase the objective function; therefore, in an optimal solution such situation is not possible.

4. Solving the multiobjective optimization problem

In general, in a multiobjective optimization problem there is no solution ω^* that minimizes all the objective functions $f_i, i = 1, \ldots, p$ simultaneously (p being the number of objectives, $p = 4$ in our problem). The goal is thus to obtain a *Pareto optimal solution* (also named *efficient* or *nondominated solution*). A feasible point ω^* is named Pareto optimal or efficient solution if there is no point that dominates it, where, in a minimization problem with p objectives, it is said that ω^1 dominates ω^2 if $f_i(\omega^1) \leq f_i(\omega^2), i = 1, \ldots, p$, and, for some $j \in \{1, \ldots, p\}$, $f_j(\omega^1) < f_j(\omega^2)$. Therefore, Pareto optimal or efficient solutions cannot improve any of the objectives without deteriorating some of the other ones. Two related concepts are those of *weakly* efficient and *strictly* efficient solutions:

- A feasible point ω^* is named *weakly* Pareto optimal or *weakly* efficient if there is no point $\omega \in \Omega$ such that $f_i(\omega) < f_i(\omega^*)$ for $i = 1, \ldots, p$.
- A feasible point ω^* is named *strictly* Pareto optimal or *strictly* efficient if there is no point $\omega \in \Omega$ such that $f_i(\omega) \leq f_i(\omega^*)$ for $i = 1, \ldots, p$.

From the above definitions, it is clear that strict efficiency implies efficiency, and efficiency implies weak efficiency.

There is a fourth type of solution, named *properly efficient solution*. These solutions are generated by the first solution approach (namely, *scalarization*) of the two outlined in Subsection 4.1. Although we will not use this approach, but the second one (namely, *lexicographic optimization*), we briefly overview properly efficiency for two reasons: (i) in case scalarization is considered in the future to solve the multiobjective CTA problem; (ii)

to show that in the multiobjective LO-CTA problem efficient solutions are also properly efficient.

Properly efficient solutions guarantee that trade-offs between objectives are bounded. Formally, a feasible point $\omega^* \in \Omega$ is properly efficient in Geoffrion sense [15] if it is efficient and there is a real number $M > 0$ such that for all $i = 1, \ldots, p$, and $\omega \in \Omega$ satisfying $f_i(\omega) < f_i(\omega^*)$ there exists an index j with $f_j(\omega^*) < f_j(\omega)$ such that

$$\frac{f_i(\omega^*) - f_i(\omega)}{f_j(\omega) - f_j(\omega^*)} \leq M. \tag{11}$$

An alternative definition of properly efficiency was provided by Kuhn and Tucker [22]. Formulating the multiobjective optimization problem as

$$\begin{array}{l} \min f(\omega) \\ \text{s. to } \omega \in \Omega = \{\omega : g(\omega) \leq 0\} \end{array} \tag{12}$$

where $f : \mathbb{R}^{n'} \to \mathbb{R}^p$ and $g : \mathbb{R}^{n'} \to \mathbb{R}^{m'}$, a feasible point $\omega^* \in \Omega$ is properly efficient in Kuhn and Tucker sense if it is efficient and there is no d satisfying:

$$\begin{array}{l} \nabla f_j(\omega^*)^\top d \leq 0 \; \forall \; j = 1, \ldots, p \\ \nabla f_i(\omega^*)^\top d < 0 \text{ for some } i \in \{1, \ldots, p\} \\ \nabla g_k(\omega^*)^\top d \leq 0 \; \forall k \in \mathcal{A}(\omega^*) = \{k \in \{1, \ldots, m\} : g_k(\omega^*) = 0\}. \end{array} \tag{13}$$

Conditions (13) mean that there is not a direction d from ω^* such that no objective locally increases, at least one strictly locally decreases, and all the points along this direction remain locally feasible. It is known that if f and g in (12) are convex and continuously differentiable, then if ω^* is properly efficient in Kuhn and Tucker sense, it is properly efficient in Geoffrion sense. We can now show that in the multiobjective LO-CTA problem all efficient solutions are properly efficient:

PROPOSITION 1 *All the efficient solutions of the multiobjective LO-CTA problem (8) are properly efficient (in Kuhn and Tucker sense, and then in Geoffrion sense).*

Proof. Problem (8) can be transformed to the form (12) (equalities $Az + \alpha^+ - \alpha^- = 0$ can be written as $Az + \alpha^+ - \alpha^- \leq 0$ and $-Az - \alpha^+ + \alpha^- \leq 0$), the feasible set Ω being a polyhedron. By (9), the four objectives f_i, $i = 1, \ldots, 4$, are linear and bounded below by 0 in the feasible region (since they are ℓ_1 norms). Since it is a linear problem, any direction d satisfying conditions (13) would be an unboundness direction for some objective $i \in \{1, \ldots, 4\}$, which is a contradiction. □

4.1 *Solution approaches*

We outline two of the most used approaches for multiobjective optimization, *scalarization* and *lexicographic optimization*:

- In the *scalarization* approach the vectorial or multiobjective function $(f_1, \ldots, f_p)$ can be transformed to a scalar function f by using a linear convex combination of the components, i.e:

$$f = \sum_{i=1}^{p} \lambda_i f_i \qquad \lambda_i \geq 0 \;\; i = 1, \ldots, p, \quad \sum_{i=1}^{p} \lambda_i = 1. \tag{15}$$

1. Let $\pi(j)$ be the preference order of $f_j, j = 1, \ldots, p$.
2. **for** $j = 1 \ldots p$ **do**
3. Solve

$$\begin{aligned} f^*_{\pi(j)} = \min_{\omega} \; & f_{\pi(j)}(\omega) \\ \text{s. to } & f_{\pi(l)}(\omega) \leq f^*_{\pi(l)} \quad l = 1, \ldots, j-1 \\ & \omega \in \Omega. \end{aligned} \tag{14}$$

4. **end for**
5. return(ω^*, solution of last optimization problem solved)

Figure 3.: The lexicographic optimization approach for (8)

The particular weights $\lambda_i, i = 1, \ldots, 4$ should be provided by the user. Let ω^* be a solution to

$$\min f(\omega) \;\; \text{s.to} \;\; \omega \in \Omega, \tag{16}$$

and $\lambda = (\lambda_1, \ldots, \lambda_p)$. It is known that (i) if $\lambda > 0$ then ω^* is a properly efficient solution; (ii) if $\lambda \geq 0$ then ω^* is a weakly efficient solution; (iii) if $\lambda \geq 0$ and ω^* is the unique solution of (16) then ω^* is a strictly efficient solution (note this would hold in the multiobjective ℓ_2-CTA problem, since the objective function of (16) would be strictly convex). In addition, all the properly and weakly efficient solutions can be obtained from some suitable λ.

In this approach is important to scale the functions f_i to guarantee that their different units can be added to a single f. We could use as scaling factor for f_i the value $\bar{f}_i = \max\{|f^*_i|, 1\}$, where f^*_i is the optimal value for this function (i.e., as if we solved the multiobjective problem considering $\lambda_i = 1$ and $\lambda_j = 0, j \neq i$). However, in practice, f^*_i is unknown (unless we solve the optimization problem using f_i as the single objective), and then some other value $f^0_i = f_i(\omega^0)$ at some (initial) feasible point ω^0 may be used. Therefore the objective function to be used in practice could be

$$f = \sum_{i=1}^{p} \lambda_i \frac{f_i}{\bar{f}_i} \qquad \lambda_i \geq 0 \;\; i = 1, \ldots, p, \quad \sum_{i=1}^{p} \lambda_i = 1. \tag{17}$$

- *Lexicographic optimization* or *lexmin optimization* considers a preference order for the different objectives, given by the permutation $\pi : \{1, \ldots, p\} \to \{1, \ldots, p\}$, $\pi(1)$ and $\pi(p)$ being respectively the objectives with highest and lowest preference. The algorithm is shown in Figure 3. This is the approach adopted in this work, since problem (8) exhibits a natural ordering of the objectives. In particular, preference should be given to feasible solutions (i.e., solutions where α^+, α^-, β_l, β_u, γ_l, and γ_u are zero), and therefore in general $\pi(4) = 1$ (i.e., $f_1(z) = ||z||_\ell$ has the lowest preference). Of the remaining three objectives in (9), f_4 should in general have the highest preference, since it controls the sensitive cells protection levels, so $\pi(1) = 4$. In this case, if no tight bounds are considered for α^+, α^-, β_l and β_u—as it was the case in all the computational tests performed—γ_l and γ_u will be 0, thus the first optimization problem is trivially solved, and protection levels are never perturbed. This will be clearly observed in the computational results section. The preference order for f_2, f_3 (that is, to give more priority to bounds or right-hand-side changes) should be decided by the user; different preferences will result in different solutions. Any solution to this lexmin problem is a Pareto or efficient solution. Given a particular preference order π, the algorithm of Figure 3 will provide a Pareto or efficient solution to (8).

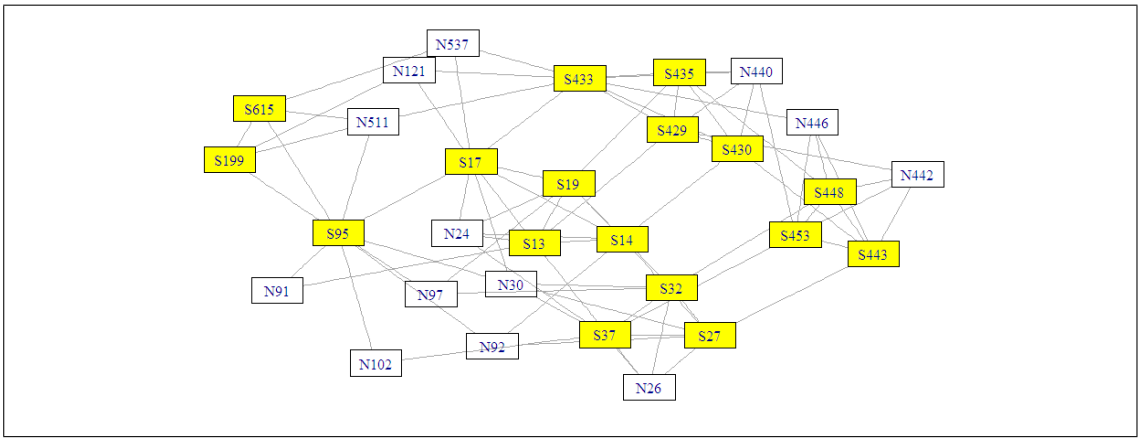

Figure 4.: The graph obtained by the network approach for instance "table7".

Given the natural ordering of the objectives, in this work we have considered lexicographic optimization. However this approach requires the solution of up to four LO problems, the last one being a challenging one, while scalarization would only involve one LO problem.

5. Fixing the binary variables

A good assignment of the binary variables is instrumental both for the feasibility and the quality of the solution provided by the lexmin approach. However, there is no direct method for an a priori good and feasible assignment for any CTA instance. Hopefully, any approximate approach should provide a good solution or, at least, to avoid solutions with severe infeasibilities in the resulting LO models.

The simplest approach is *random* assignment. Sensitive cells are oriented either to positive or negative directions with equal probabilities. Although randomness is fairly good at dividing the sensitive cells between both directions, this method cannot take into account any specific feature of the table.

The SAT heuristic (which was developed in [17] for the solution of the MILO-CTA problem by a block coordinate descent heuristic) can be useful to obtain a wiser proposal. Directions of sensitive cells are computed by solving a satisfiability (SAT) problem on rules derived from the table constraints (see [17] for details). Nevertheless, some cells could be undetermined if no rule is indicating a given direction, and these cells should be assigned at random. The SAT approach (as the below network and both SAT-network approaches) cannot guarantee a feasible solution of (2). However they help in reducing the level of infeasibility, compared to the previously discussed random assignment, such that components f_2, f_3 and f_4 of (8) will have a lower value in the optimal solution.

As for the SAT approach, the next new alternative developed in this work is based also in the constraints structure of the problem, and neither guarantees a feasible solution. This method is called *network* because a graph is constructed from the set of constraints of the CTA problem. An example of such a graph can be seen in Figure 4, obtained from the real instance "table7", one of the test cases from Table 1 to be used in the computational results of Section 6. In this graph the vertices represent cells and the edges constraints. Specifically, the graph is built in two steps:

(1) First, we create a vertex for every sensitive cell, and an edge between every pair of vertices such that there is a constraint containing both cells.
(2) Second, we add a vertex for every non-sensitive cell appearing at least in two constraints together with a sensitive cell.

In Figure 4, sensitive and non-sensitive cells have been represented by yellow and white boxes, with labels started by "S" and "N", respectively. In both cases, the number in the label corresponds to the cell number for this particular instance.

Each vertex may have a direction $+1$ or -1, and we start with a graph where all the vertices are undefined (say, 0). The goal is to assign a direction to each vertex so that the number of mismatches is the minimum. In a broad sense, we call *mismatch* a situation where both nodes of some edge are contributing to the constraint imbalance in the same direction. For better understanding, consider the structure of general constraints of a table:

$$Az = 0 \Leftrightarrow a_{i,j_1} z_{j_1} + a_{i,j_2} z_{j_2} + \cdots + a_{i,j_k} z_{j_k} = 0 \quad i \in \mathcal{M}.$$

Usually, only a few cells are involved in the i-th constraint (say, k cells), such that $k-1$ coefficients are 1 and just one is -1, meaning that the sum of deviations z must be equal to the marginal deviation. So, two cells linked by an edge are matched when:

- both point upwards and one of them has a negative coefficient;
- both point downwards and one of them has a negative coefficient;
- they point to different directions and have positive coefficients.

The main algorithm of the network method attempts to find the most suitable orientation to each sensitive cell, trying to match as many edges as possible. So, starting from a given vertex, it traverses the graph in an order defined from its topology (depending on the first vertex, the order may be different, so the assignment would differ as well).

Since a cell may appear in several constraints, conflicting situations appear often, as long as any assignment to the current vertex would cause a mismatch in some of the incident edges. In such situations the heuristic implemented chooses the direction that minimizes the number of mismatches, or a random choice in case of a tie.

Finally, the heuristic repeats the main algorithm from many starting points, computing a global index of mismatch for the assignment. The procedure returns the assignment with the least index.

A hybrid between the SAT and network methods is also possible: the suggested directions given by the SAT output are fixed and not assigned within the network process. This fourth approach will be named *both SAT-network*.

The above four approaches (random, SAT, network and both SAT-network) to fix the protection directions of sensitive cells are evaluated in Section 6.4.

6. Computational results

The lexicographic approach based on LO-CTA has been implemented within the FP7-INFRA-2010-262608 "Data without Boundaries" European Union project. This code has been included in the τ-Argus package since version 4.1.0 [13, 19] (which can be freely obtained from `http://neon.vb.cbs.nl/casc/tau.htm`), used by European (and some non-European) statistical agencies. The code may use up to five different solvers, both commercial and open source. In this work we will only provide results with the commercial solvers CPLEX and XPRESS, and the open source solver Clp [14] from the COIN-OR project. All the runs were carried out on a Fujitsu Primergy RX300 server with two 3.33 GHz Intel Xeon X5680 CPUs (each CPU with 12 cores) and 144 GB of RAM, under a GNU/Linux operating system (Suse 11.4), without exploitation of multithreading capabilities.

We have considered a set of 38 standard instances in the literature on statistical disclosure control on tabular data [4, 6, 18]. Most of them are publicly available real tables

instance	n	s	m	n. coef
australia_ABS	24420	918	274	13224
bts4	36570	2260	36310	136912
cbs	11163	2467	244	22326
dale	16514	4923	405	33028
destatis	5940	621	1464	18180
five20b	34552	3662	52983	208335
five20c	34501	4022	58825	231345
hier13	2020	112	3313	11929
hier13x13x13a	2197	108	3549	11661
hier13x13x13b	2197	108	3549	11661
hier13x13x13c	2197	108	3549	11661
hier13x13x13d	2197	108	3549	11661
hier13x13x13e	2197	112	3549	11661
hier13x13x7d	1183	75	1443	5369
hier13x7x7d	637	50	525	2401
hier16	3564	224	5484	19996
hier16x16x16a	4096	224	5376	21504
hier16x16x16b	4096	224	5376	21504
hier16x16x16c	4096	224	5376	21504
hier16x16x16d	4096	224	5376	21504
hier16x16x16e	4096	224	5376	21504
nine12	10399	1178	11362	52624
nine5d	10733	1661	17295	58135
ninenew	6546	858	7340	32920
osorio	10201	7	202	20402
sbs2008_C	4212	1135	2580	13806
sbs2008_D_b	28288	7131	13360	87022
sbs2008_E	1430	382	991	4680
table1	1584	146	510	4752
table3	4992	517	2464	19968
table4	4992	517	2464	19968
table5	4992	517	2464	19968
table6	1584	146	510	4752
table7	624	17	230	1872
table8	1271	3	72	2542
targus	162	13	63	360
toy3dsarah	2890	376	1649	9690
two5in6	5681	720	9629	34310

Table 1. Dimensions of instances.

generated by NSAs, and only a few of them are confidential tables generated by Eurostat (instances "sbs*")—the statistical agency of the European Commission, and Destatis (instance "destatis")—the German NSA. Table 1 reports the main characteristics of these instances: number of cells (column n), number of sensitive cells (column s), number of constraints (column m), and number of nonzero coefficients in the constraints matrix A defining the table relations.

To avoid errors due to numerical tolerances, in the implementation developed constraints $f_{\pi(l)}(\omega) \leq f^*_{\pi(l)}$ of (14) were replaced by $f_{\pi(l)}(\omega) \leq f^*_{\pi(l)}(1+\epsilon_f)$, ϵ_f being a small value (e.g., 10^{-4}). Using a value $\epsilon_f = 0$ resulted in larger solution times, and, very often, in infeasibility issues.

The first four following subsections report the results obtained in the evaluation of priority orders, solvers, percentages of deviations in cell values, and procedures for fixing the protection directions of sensitive cells with LO-CTA. The fifth subsection provides a comparison between LO-CTA and MILO-CTA in terms of solution time and quality of the solution.

6.1 *Evaluation of priority orders*

Table 2 shows the results for the evaluation of priority orders $\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$ (where $\pi = (\pi(1), \ldots, \pi(4))$. For each instance the table reports the CPU time of the lexmin optimization, and the optimal value of each objective function (columns f_i, $i = 1, \ldots, 4$). The fastest run is marked in boldface. These executions were performed with the standard (i.e., the infeasible primal-dual path-following) barrier algorithm of CPLEX, randomly fixing the protection directions of sensitive cells (but using the same

instance	CPU time		f_1		f_2		f_3	
order:	(4,3,2,1)	(4,2,3,1)	(4,3,2,1)	(4,2,3,1)	(4,3,2,1)	(4,2,3,1)	(4,3,2,1)	(4,2,3,1)
australia_ABS	**2.99**	3.01	10471.8	52313.6	4138.34	16.3086	0.388728	397.967
bts4	**731.89**	3429.09	4.64199e+09	6.0883e+09	1896.88	1582.02	0	289.935
cbs	**0.25**	0.37	1.14784e+06	2.70038e+06	35.8425	15.9618	0	19.8191
dale	**0.52**	1.3	494	494	0	0	0	0
destatis	**0.94**	2.66	1.08813e+09	1.32972e+09	1220.13	528.131	0	3440.32
five20b	> 3600	> 3600	—	—	6760.94	5741.95	0	1653.35
five20c	> 3600	> 3600	—	—	8266.2	7073.67	0	2544.33
hier13	**3.07**	8.48	3.9942e+08	6.24203e+08	397.596	362.27	0	34.4359
hier13x13x13a	**2.79**	25.55	4.86898e+08	7.85047e+08	378.277	340.364	0	40.5799
hier13x13x13b	**4.09**	21.04	40818.9	56853.2	378.277	340.364	0	40.5799
hier13x13x13c	**5.06**	20.8	342477	482136	378.277	340.364	0	40.5799
hier13x13x13d	**4.71**	14.81	476984	848912	621.512	510.546	0	140.891
hier13x13x13e	**3.67**	15.63	4.50019e+06	1.06078e+07	661.835	562.487	0	141.853
hier13x13x7d	**0.94**	1.83	1.86068e+06	2.05762e+06	152.903	145.978	0	9.50125
hier13x7x7d	**0.22**	0.34	768054	832149	62.1286	60.581	0	1.45358
hier16	**31.27**	822.08	8.03562e+08	1.05558e+09	332.65	299.565	0	22.1156
hier16x16x16a	**10.2**	114.35	8.74132e+08	1.17814e+09	536.215	490.13	0	29.46
hier16x16x16b	**14.68**	112.43	81238.6	100391	536.215	490.13	0	29.46
hier16x16x16c	**18.45**	68.24	675679	847619	536.215	490.13	0	29.46
hier16x16x16d	**11.4**	95.7	1.30933e+09	2.40793e+09	853.206	735.2	0	101.305
hier16x16x16e	**19.07**	61.83	1.08077e+07	1.65291e+07	853.206	735.2	0	101.305
nine12	**29.33**	784.24	1.88278e+09	2.74804e+09	989.62	683.641	0	324.934
nine5d	239.13	**68**	1.22666e+09	1.69348e+09	3778.88	3649.69	0	259.64
ninenew	**21.54**	60.89	1.27819e+09	1.98047e+09	941.067	662.144	0	434.903
osorio	**0.3**	1.88	15	15	0	0	0	0
sbs2008_C	**0.31**	0.5	424452	581521	29856.7	28152	83.6344	8210.57
sbs2008_D_b	**2.31**	5.41	778510	1.5701e+06	83356	77514.7	107.658	16664.5
sbs2008_E	**0.08**	0.12	105868	302176	11923	9984.73	6.87337	4915.61
table1	**0.1**	0.24	7.71857e+13	9.24447e+13	545.123	288.929	0	179.689
table3	**1.04**	6.5	5.30633e+12	3.68048e+13	1383.19	570.329	0	5865.61
table4	**1.03**	3.77	4.47093e+10	3.10105e+11	1383.19	570.329	0	5865.61
table5	**1.04**	3.25	4.44374e+07	3.09593e+08	1383.19	570.329	0	5865.61
table6	**0.08**	0.19	7.57471e+10	9.07203e+10	534.544	278.56	0	179.544
table7	**0.03**	0.16	4.58472e+09	1.44823e+11	298.144	4.57506	0	827.618
table8	**0.04**	0.07	725	725	0	0	0	0
targus	0.02	**0.01**	1.98165e+06	1.97917e+06	0.309239	0.114767	0	0.197297
toy3dsarah	**0.15**	0.34	6.37301e+14	6.825e+14	690816	657301	0	1241450
two5in6	**13.82**	145.82	7.38413e+08	9.85666e+08	1693.99	1590.61	0	116.905

— objective value not computed by time limit reached

Table 2. Results with priority orders $\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$, using barrier CPLEX as solver, allowing a 2% of deviations in cell values, and the random heuristic to fix the protection direction of sensitive cells. In boldface, the fastest run. f_4 is not shown since it was 0 in all instances for both orders.

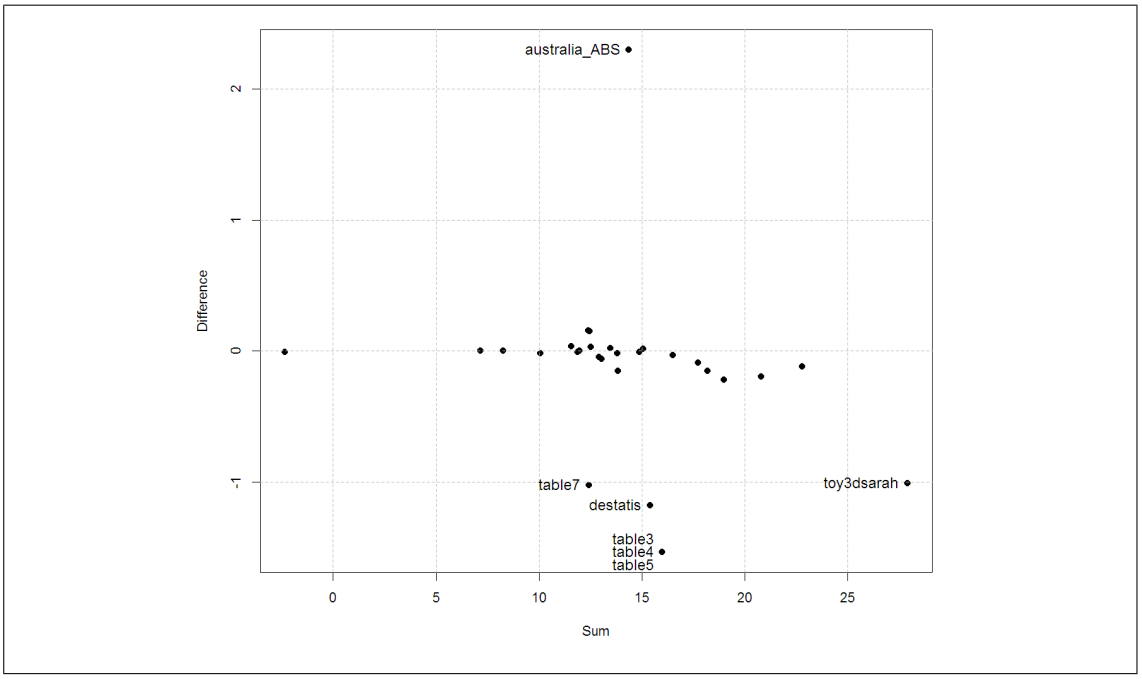

Figure 5.: Plot of Sum= $\log\left(f_2^{(4,3,2,1)} + f_3^{(4,3,2,1)}\right) + \log\left(f_2^{(4,2,3,1)} + f_3^{(4,2,3,1)}\right)$ vs Difference= $\log\left(f_2^{(4,3,2,1)} + f_3^{(4,3,2,1)}\right) - \log\left(f_2^{(4,2,3,1)} + f_3^{(4,2,3,1)}\right)$, for every instance.

random directions for the two priority orders), and allowing a 2% of deviations in the cells values. This 2% maximum deviation in cell values is a practical requirement of NSAs (in particular it was told to us by Destatis, the German NSA), in a attempt to maximize the utility of the resulting table, making the optimization problem significantly more difficult. The time limit of 3600 was reached for two instances, whose value for f_1 is marked with "—" (the other problems could be solved within the time limit).

Objective f_4 is not shown since it was 0 in all instances for both orders (as it was discussed in Subsection 4.1). This means (quite obviously) that we do not need to change the protection levels if bounds and constraints can be modified. For order $\pi = (4, 3, 2, 1)$ in all the instances, but four, bounds were not modified (that is, $f_3 = 0$), which forced significant changes in constraints (that is, f_2 is large). For order $\pi = (4, 2, 3, 1)$ both f_2 and f_3 were positive, but the changes in constraints were not so large. If we focus on the norm of the cell deviations, f_1 was much less for $\pi = (4, 3, 2, 1)$, and this also happened for the CPU time (only in two instances the order $\pi = (4, 2, 3, 1)$ provided a fastest execution). At a first glance, it seems there is no clear winner: on the one hand, according to f_1 and the CPU time, it could be concluded that $\pi = (4, 3, 2, 1)$ is preferable; on the other hand, this order involves larger modifications in constraints due to f_2. Both orders provide efficient or Pareto solutions, and depending on their different properties, NSAs may prefer one order or the other. Figure 5 gives us more insight into the behaviour of both orders. The horizontal axis of that figure is related to the sum $\log\left(f_2^{(4,3,2,1)} + f_3^{(4,3,2,1)}\right) + \log\left(f_2^{(4,2,3,1)} + f_3^{(4,2,3,1)}\right)$, while the vertical axis is associated with the difference $\log\left(f_2^{(4,3,2,1)} + f_3^{(4,3,2,1)}\right) - \log\left(f_2^{(4,2,3,1)} + f_3^{(4,2,3,1)}\right)$; each instance is represented by a point in the figure. Note that it makes sense to consider $f_2 + f_3$, since f_2 represents changes in constraints (that is, deviations from the value of the marginal cell, since nonzero values in matrix A are only $+1$ or -1), and f_3 changes in cell bounds. From the figure, we see that the differences are around zero for most instances, that is, the amount of constraints and bounds changes $f_2 + f_3$ was similar for both orders. The only

exceptions were "australia_ABS" (in which order $\pi = (4, 3, 2, 1)$ provided a larger f_2+f_3), and "destatis", "toy3dsarah", "table3", "table4", "table5" and "table7" (order $\pi = (4, 2, 3, 1)$ provided the larger changes in those cases). Therefore, from Figure 5 it cannot be stated that the less CPU time and value of f_1 provided by $\pi = (4, 3, 2, 1)$ are due to this order providing larger changes. According to this, order $\pi = (4, 3, 2, 1)$ could be considered a better choice.

For the computational results of the following Subsections 6.2–6.4 we will consider the order $\pi = (4, 2, 3, 1)$: being more time consuming, it can be considered the worst case, thus the preferred option for a fair evaluation of LO-CTA. Both priority orders will be again considered in Subsection 6.5, in which LO-CTA is compared to MILO-CTA.

6.2 *Evaluation of solvers*

Table 3 shows the CPU time with the interior-point (columns "barrier") and simplex algorithms of the commercial solvers CPLEX and XPRESS, and the open source Clp solver from the COIN-OR project. These executions were performed with priority order $\pi = (4, 2, 3, 1)$, allowing a 2% of deviations in cell values, and the random heuristic to fix the protection direction of sensitive cells. The CPU time of the fastest algorithm (interior-point or simplex) for each solver is marked in italics; in addition, the fastest run among all the solvers is in boldface. Problems marked with "—" for Clp were reported as infeasible (though it is feasible). The time limit of 3600 seconds was reached for several instances, thus no efficient solution was computed in those situations.

The immediate conclusion from the results of Table 3 is that interior-point algorithms are more efficient than simplex in this kind of application, mainly for the two commercial solvers (it is known than the interior-point of Clp is not competitive against commercial implementations, unlike its simplex code which is highly efficient). This is consistent with previous works on CTA [4, 6, 18]. This is specially relevant in some of the largest instances, e.g., "nine5d", where the barrier was much faster than simplex. We also see there are significant differences between solvers; for example, instance "table4" took 150.91 seconds with the simplex of CPLEX, only 6.85 with that of XPRESS, and 627 with Clp; but we got opposite results for instance "nine12". Another conclusion is that, for these LO-CTA problems, the barrier of XPRESS, which is based on the BPMPD solver [23], seems to be a very efficient option.

It was also observed that problems (14) within the lexmin optimization became more difficult for the barrier algorithm as new constraints $f_{\pi(l)}(\omega) \leq f^*_{\pi(l)}$ were added. This is illustrated in Table 4, which reports for the subset of largest instances the CPU time and number of iterations (in brackets) of the interior-point and simplex algorithms of XPRESS for the four subproblems solved (using the order $\pi = (4, 2, 3, 1)$). The interior-point CPU time does not include the crossover (that is, the procedure for obtaining a vertex solution from the (possibly non-vertex) primal-dual solution computed by the interior-point algorithm); this explains why the sum of the four times does not equal the values in Table 3. Indeed, it can be seen that instances "five20b" and "five20c" exhausted the time limit because of the crossover. (Crossover can be deactivated in the application developed, but in general it provides sparser solutions—since they are basic solutions, unlike the interior-point ones—which are preferred.) It is clearly seen that as new constraints of the type $f_{\pi(l)}(\omega) \leq f^*_{\pi(l)}$ are added the number of interior-point iterations (and its CPU time) increases significantly. However this fact is not observed for the simplex method. Although we did not analyze in detail this different behaviour, a possible explanation would be that the feasible region is significantly reduced as we are adding new constraints (we limit the feasible region of the new problem to be the optimal face of the previous one). The central-path of the new problem, which the path-following methods implemented in

	CPLEX		XPRESS		Clp	
instance	barrier	simplex	barrier	simplex	barrier	simplex
australia_ABS.csp	3.01	***1.05***	2.28	*1.52*	2.25	*2.24*
bts4.csp	*3429.09*	> 3600	> 3600	> 3600	***1273.29***	—
cbs.csp	***0.37***	0.65	0.6	*0.48*	0.52	*0.5*
dale.csp	***1.3***	10.13	*1.63*	1.84	1.89	*1.8*
destatis.csp	***2.66***	30.43	*3.43*	5.35	13	*9.08*
five20b.csp	> 3600	> 3600	> 3600	> 3600	> 3600	> 3600
five20c.csp	> 3600	> 3600	> 3600	> 3600	> 3600	> 3600
hier13.csp	***8.48***	39.91	*11.64*	27.76	*47.56*	—
hier13x13x13a.csp	*25.55*	53.61	***7.11***	21.48	276.51	*49.23*
hier13x13x13b.csp	*21.04*	29.37	***11.25***	23.08	271.31	*48.61*
hier13x13x13c.csp	*20.8*	67.58	***6.58***	22.92	306.3	*48.61*
hier13x13x13d.csp	*14.81*	72.88	***12.43***	28.65	*75.55*	—
hier13x13x13e.csp	*15.63*	45.46	***9.8***	24.73	*59.5*	—
hier13x13x7d.csp	*1.83*	3.24	***1.14***	2.81	2.06	*1.69*
hier13x7x7d.csp	0.34	*0.26*	0.32	*0.29*	0.24	***0.22***
hier16.csp	*822.08*	1150.16	***112.79***	287.47	> 3600	—
hier16x16x16a.csp	*114.35*	240.82	***33.11***	196.05	*374.96*	—
hier16x16x16b.csp	*112.43*	223	***31.92***	120.15	*429*	—
hier16x16x16c.csp	*68.24*	285.45	***37.91***	127.1	*385.2*	—
hier16x16x16d.csp	*95.7*	179.76	***32.07***	106.36	*464.43*	—
hier16x16x16e.csp	*61.83*	291.04	***32.31***	99.05	*344*	—
nine12.csp	*784.24*	1618.35	> 3600	*2740.72*	928.18	***713.59***
nine5d.csp	***68***	2049.09	*130.46*	1363.12	> 3600	*412.81*
ninenew.csp	***60.89***	171.7	*62.41*	195.86	159.05	*116.14*
osorio.csp	1.88	***0.57***	*1.92*	2.84	1.8	*1.75*
sbs2008_C.csp	*0.5*	0.88	0.57	***0.26***	2.12	*1.17*
sbs2008_D_b.csp	*5.41*	12.03	4.31	***1.73***	27.34	*25.14*
sbs2008_E.csp	0.12	***0.08***	0.19	*0.15*	0.13	*0.1*
table1.csp	***0.24***	0.51	*0.34*	0.51	0.58	*0.41*
table3.csp	*6.5*	52.71	***4.86***	6.6	*170.97*	944.8
table4.csp	***3.77***	150.91	*4.8*	6.85	*167.31*	627.86
table5.csp	***3.25***	49.15	*4.93*	6.63	*165.09*	510.19
table6.csp	***0.19***	0.55	*0.36*	0.52	*0.39*	0.5
table7.csp	0.16	*0.11*	0.21	*0.16*	0.14	***0.1***
table8.csp	*0.07*	*0.07*	*0.18*	*0.18*	***0.03***	0.04
targus.csp	***0.01***	***0.01***	*0.09*	*0.09*	***0.01***	0.02
toy3dsarah.csp	*0.34*	1.52	0.47	***0.22***	6.63	*0.84*
two5in6.csp	*145.82*	237.9	***47.01***	185.44	210.82	*103.86*

— problem reported as infeasible (though it is feasible)

Table 3. CPU time with interior-point and simplex algorithms of solvers CPLEX, XPRESS and Clp, using order $\pi = (4, 2, 3, 1)$, allowing a 2% of deviations in cell values, and the random heuristic to fix the protection direction of sensitive cells. In italics the fastest run between interior-point and simplex for each solver; in italic and boldface, the fastest run for all the solvers.

	barrier				simplex			
instance	f_4	f_2	f_3	f_1	f_4	f_2	f_3	f_1
bts4	0 (0)	11 (23)	23 (51)	31 (85)	0(0)	145 (255K)	1167 (542K)	—
five20b	0 (0)	316 (24)	—	—	0(0)	—	—	—
five20b	0 (0)	475 (30)	—	—	0(0)	—	—	—
hier16	0 (0)	8 (20)	23 (37)	30 (70)	0(0)	38 (85K)	145 (116K)	104 (70K)
nine5d	0 (0)	19 (26)	17 (38)	57 (84)	0(0)	107 (222K)	728 (310K)	635 (147K)
nine12	0 (0)	36 (29)	56 (49)	106 (101)	0(0)	277 (280K)	1982 (577K)	481 (129K)
two5in6	0 (0)	13 (36)	10 (48)	21 (76)	0(0)	36 (126K)	59 (49K)	90 (53K)

— time limit reached

Table 4. CPU time and number of iterations (within brackets, in thousands for simplex) of XPRESS with interior-point and simplex algorithms for the solution of each particular objective in the largest instances. Barrier time does not include crossover (this explains why the time limit of 3600 seconds is not reached in instances bts4 and nine12, unlike in Table 3).

CPLEX, XPRESS and Clp attempt to "follow", is thus confined to a much smaller region, such that large directions can not be taken by the interior-point algorithm. This would explain the unexpectedly large number of interior-point iterations of the last subproblem with f_1. On the other hand, this reduction in the size of the feasible region may not negatively affect to simplex; indeed, reducing the number of feasible vertices may even be a benefit, as shown in some instances of Table 4.

instance	CPU time				f_2			
perc.:	2%	5%	10%	100%	2%	5%	10%	100%
australia_ABS	2.25	**1.97**	2.21	†	16.31	16.31	16.31	†
bts4	>3600	1628	1052	**821**	1582	1582	1582	1582
cbs	0.6	0.6	0.6	**0.53**	15.96	15.96	15.96	15.96
dale	1.63	1.54	1.5	**1.35**	0	0	0	0
destatis	3.43	**3.16**	3.88	4.19	528	528	528	528
five20b	>3600	>3600	>3600	>3600	0	0	0	0
five20c	>3600	>3600	>3600	>3600	0	0	0	0
hier13	11.6	9.83	10.03	**6.58**	362	362	362	362
hier13x13x13a	7.12	6.1	**4.45**	5.09	340	340	340	340
hier13x13x13b	11.3	5.14	**3.48**	4.6	340	340	340	340
hier13x13x13c	6.58	6.28	**4.06**	4.34	340	340	340	340
hier13x13x13d	12.4	7.27	6.4	**3.88**	511	511	511	511
hier13x13x13e	9.8	14.44	6.41	**4.79**	562	562	562	562
hier13x13x7d	1.15	1.72	0.81	**0.77**	146	146	146	146
hier13x7x7d	0.32	0.28	**0.27**	**0.27**	60.58	60.58	60.58	60.58
hier16	113	49.51	**26.99**	35.88	300	300	300	300
hier16x16x16a	33.08	28.97	**20.54**	20.88	490	490	490	490
hier16x16x16b	31.85	36.4	**20.38**	20.85	490	490	490	490
hier16x16x16c	37.94	29.69	**21.08**	22.41	490	490	490	490
hier16x16x16d	32.04	24.95	25.94	**23.13**	735	735	735	735
hier16x16x16e	32.16	20.34	21.64	**20.11**	735	735	735	735
nine12	>3600	187	207	**178**	684	684	684	684
nine5d	131	99.33	116	**74.59**	3650	3650	3650	3650
ninenew	62.31	76.02	**54.6**	68.61	662	662	662	662
osorio	1.91	1.92	1.93	**1.38**	0	0	0	0
sbs2008_C	**0.57**	**0.57**	**0.57**	†	28152	28152	28152	†
sbs2008_D_b	4.32	**4.06**	4.56	†	77515	77515	77515	†
sbs2008_E	**0.18**	0.21	0.21	†	9985	9985	9985	†
table1	0.36	**0.28**	0.31	0.35	289	289	289	289
table3	4.86	3.87	4.06	**2.53**	570	570	570	570
table4	4.83	3.49	5.61	**2.4**	570	570	570	570
table5	4.92	3.38	4.2	**2.49**	570	570	570	570
table6	0.37	0.34	**0.3**	0.59	279	279	279	279
table7	0.21	0.21	0.19	**0.18**	4.575	4.575	4.575	4.575
table8	0.21	0.19	0.22	**0.15**	0	0	0	0
targus	0.1	**0.08**	0.1	0.1	0.1148	0.1148	0.1148	0.1148
toy3dsarah	**0.48**	0.49	**0.48**	0.55	657301	657301	657301	657301
two5in6	**47.01**	47.16	56.11	50.63	1591	1591	1591	1591

† problem reported as infeasible

Table 5. CPU time and value of f_2 for different maximum percentages of deviation in cell values, using the interior-point of XPRESS as solver, order $\pi = (4, 2, 3, 1)$, and the random heuristic to fix the protection direction of sensitive cells. In boldface, the fastest run.

6.3 *Evaluation of allowed percentage of deviations in cell values*

Tables 5 and 6 report the CPU time of the lexmin optimization, and the optimal value of objective functions f_1, f_2 and f_3, allowing different percentage of deviations in cells values (2%, 5%, 10% and 100%). Objective f_4 is not reported since it was 0 in all the executions (as discussed above). The order $\pi = (4, 2, 3, 1)$ and the interior-point of XPRESS were used in these runs. The fastest execution is marked in boldface. A time limit of 3600 seconds was also used; when the time limit is reached tables 5 and 6 report the best objective function found so far. Notice that f_2 reached the same value, independently of the percentage, for every instance. As expected, the problems become generally easier when the percentage of allowed deviations increases, and most of the fastest executions are obtained with 10% and 100%. However, and unexpectedly, the real problems by Eurostat "sbs2008_C", "sbs2008_D_b" and "sbs2008_E", and "australia_ABS" instance, were reported as infesible with 100%, while they could be easily solved with 2%, 5% and 10%. As for the objective functions, there are not significant differences, since they report aggregated information for all the cells. However, in practice, using a tight percentage of 2% (which is the default value considered) provides protected tables where all the cells are consistently close to those of the original table.

instance		f_3				f_1			
	perc.:	2%	5%	10%	100%	2%	5%	10%	100%
australia_ABS		398	232	100	†	52314	46365	44028	†
bts4		290	68.59	27.27	7.395	0	4.376e+09	3.89e+09	3.549e+09
cbs		19.82	12.51	6.633	0	2.7e+06	1.93e+06	1.073e+06	393187
dale		0	0	0	0	494	494	493	460
destatis		3440	2982	2851	2209	1.33e+09	6.618e+08	5.357e+08	2.912e+08
five20b		—	—	—	—	—	—	—	—
five20c		—	—	—	—	—	—	—	—
hier13		34.44	7.213	2.145	0	6.242e+08	5.372e+08	5.273e+08	5.211e+08
hier13x13x13a		40.58	4.586	0	0	7.85e+08	7.066e+08	6.66e+08	6.537e+08
hier13x13x13b		40.58	4.586	0	0	56853	56060	54723	54056
hier13x13x13c		40.58	4.586	0	0	482136	477166	463534	457194
hier13x13x13d		141	45	7.178	0	848912	805797	792752	761933
hier13x13x13e		142	45.69	12.19	0	1.061e+07	9.651e+06	9.641e+06	9.436e+06
hier13x13x7d		9.501	0.4985	0	0	2.058e+06	1.952e+06	1.897e+06	1.846e+06
hier13x7x7d		1.454	0	0	0	832149	760481	760107	760107
hier16		22.12	1.254	0	0	1.056e+09	8.173e+08	7.715e+08	7.595e+08
hier16x16x16a		29.46	2.556	0	0	1.178e+09	9.504e+08	8.783e+08	8.491e+08
hier16x16x16b		29.46	2.556	0	0	100391	96351	94685	94622
hier16x16x16c		29.46	2.556	0	0	847619	802968	785441	784120
hier16x16x16d		101	18.04	1.174	0	2.408e+09	1.564e+09	1.42e+09	1.313e+09
hier16x16x16e		101	18.04	1.174	0	1.653e+07	1.474e+07	1.449e+07	1.426e+07
nine12		325	71.67	31.34	3.84	0	1.677e+09	1.513e+09	1.373e+09
nine5d		260	78.59	42.87	0	1.693e+09	1.251e+09	1.187e+09	1.123e+09
ninenew		435	129	51.07	6.273	1.98e+09	1.275e+09	1.095e+09	9.62e+08
osorio		0	0	0	0	15	15	15	15
sbs2008_C		8211	7309	6430	†	581521	582320	582643	†
sbs2008_D_b		16664	13107	10422	†	1.57e+06	1.77e+06	1.587e+06	†
sbs2008_E		4916	3982	3369	†	302176	302085	301956	†
table1		180	0	0	0	9.244e+13	6.118e+13	4.652e+13	3.371e+13
table3		5866	5329	4987	3495	3.68e+13	3.04e+13	2.592e+13	1.174e+13
table4		5866	5329	4987	3495	3.101e+11	2.562e+11	2.184e+11	9.89e+10
table5		5866	5329	4987	3495	3.096e+08	2.557e+08	2.178e+08	9.831e+07
table6		180	0	0	0	9.072e+10	6.003e+10	4.565e+10	3.308e+10
table7		828	778	705	538	1.448e+11	9.892e+10	8.112e+10	1.119e+11
table8		0	0	0	0	725	642	586	484
targus		0.1973	0.1807	0.164	0.1446	1.979e+06	1.167e+06	1.135e+06	1.077e+06
toy3dsarah		1.241e+06	1.159e+06	1.069e+06	738090	6.825e+14	6.737e+14	6.605e+14	5.739e+14
two5in6		117	31.94	20.39	9.744	9.857e+08	8.204e+08	7.798e+08	7.478e+08

† problem reported as infeasible
— objective value not computed by time limit reached

Table 6. Values of f_3 and f_1 for different maximum percentages of deviation in cell values, using the interior-point of XPRESS as solver, order $\pi = (4, 2, 3, 1)$, and the random heuristic to fix the protection direction of sensitive cells.

instance	CPU time				f_2			
method:	random	SAT	network	both	random	SAT	network	both
australia_ABS	**2.66**	3.23	*	12.88	3.805	**0**	—	**0**
bts4	> 3600	3188	3309	**3157**	1582	42.02	534	**41.47**
cbs	**0.44**	2.73	*	2176	15.96	**0**	—	**0**
dale	**1.33**	5.6	*	*	**0**	**0**	—	—
destatis	4.32	**1.87**	6.75	4.1	528	**0**	743	**0**
five20b	> 3600	> 3600	> 3600	> 3600	5742	26.78	2994	**0**
five20c	> 3600	> 3600	> 3600	> 3600	7074	24.73	3880	**21.43**
hier13	12.17	**8.64**	22.23	10.24	362	64.73	85.03	**45.25**
hier13x13x13a	9.47	**1.91**	9.66	2.18	340	**0**	242	**0**
hier13x13x13b	9.68	**1.85**	9.67	2.03	340	**0**	242	**0**
hier13x13x13c	10.82	**1.92**	8.49	2.04	340	**0**	242	**0**
hier13x13x13d	11.39	**1.92**	11.53	2.04	511	**0**	362	**0**
hier13x13x13e	18.42	**1.9**	12.28	1.95	562	**0**	367	**0**
hier13x13x7d	1.34	0.38	1.3	**0.36**	146	**0**	174	**0**
hier13x7x7d	0.26	**0.11**	0.37	0.12	60.58	**0**	88.72	**0**
hier16	58.66	64.82	102	**32.45**	300	77.43	206	**39.7**
hier16x16x16a	68.59	**6.7**	85.49	7.26	490	**0**	565	**0**
hier16x16x16b	40.84	**6.29**	49.96	6.93	490	**0**	565	**0**
hier16x16x16c	38.92	**6.58**	45.1	7.03	490	**0**	565	**0**
hier16x16x16d	48.59	**7.14**	57.84	7.28	735	**0**	850	**0**
hier16x16x16e	47.05	**6.6**	60.58	7.14	735	**0**	850	**0**
nine12	210	802	734	**33.02**	684	27.73	448	**0**
nine5d	477	73.61	703	**17.51**	3650	13.88	1782	**0**
ninenew	195	141	181	**19.32**	662	54.43	434	**0**
osorio	**1.74**	**1.74**	1.78	1.77	**0**	**0**	**0**	**0**
sbs2008_C	**0.53**	0.68	2.77	1.23	28156	**1115**	27808	**1115**
sbs2008_D_b	**6.26**	7.79	200	38.17	77899	**7039**	66104	7075
sbs2008_E	**0.13**	**0.13**	0.3	0.17	9990	**135**	11108	**135**
table1	0.34	**0.2**	0.43	0.24	289	**0**	834	**0**
table3	5.11	3.23	11.58	**3.05**	570	**9.848**	561	**9.848**
table4	5.05	**2.75**	5.74	3.14	570	**9.848**	561	**9.848**
table5	4.69	2.96	7.21	**2.72**	570	**9.848**	561	**9.848**
table6	0.27	**0.18**	0.38	0.21	279	**0**	814	**0**
table7	0.15	**0.08**	0.14	**0.08**	4.575	**0**	0.6003	**0**
table8	0.09	**0.08**	**0.08**	0.09	**0**	**0**	**0**	**0**
targus	**0.03**	**0.03**	**0.03**	**0.03**	0.1148	**0**	**0**	**0**
toy3dsarah	0.49	**0.27**	0.85	0.32	657301	**0**	455360	**0**
two5in6	66.01	**7.51**	175	7.91	1591	**0**	847	**0**

* heuristic failed
— objective value not computed, either by time limit reached or by heuristic failure

Table 7. CPU time and value of f_2 with the four heuristics for fixing the protection direction of sensitive cells, using the interior-point of CPLEX as solver, order $\pi = (4, 2, 3, 1)$, and allowing a 2% of deviations in cell values. In boldface, the run with the best quality solution.

6.4 *Evaluation of procedures to fix protection directions of sensitive cells*

Tables 7 and 8 report the CPU time of the lexmin optimization and the optimal value of each objective function (excluding f_4, which is not reported since it was 0 in all the runs), for the four different heuristics described in Section 5 to fix the protection directions of sensitive cells. The CPLEX homogeneous self-dual interior-point algorithm was used in these runs, instead of the standard infeasible path-following variant of Subsections 6.1 and Subsections 6.2; this explains the differences in CPU times of Tables 2–3 vs. Table 7 for CPLEX barrier and the random heuristic. As it will be shown in below Subsection 6.5, although the standard barrier algorithm is usually considered the fastest option, it was outperformed in some of the difficult instances by the slower—but numerically more stable when the problem is near-infeasible—homogeneous self-dual variant. The best solution is marked in boldface, according to the lexicographic order $\pi = (4, 2, 3, 1)$ considered in those runs. A time limit of 3600 seconds was considered, which was reached in some executions. The network heuristic failed in some runs, clearly marked in the table.

From Tables 7 and 8 we see that, according to the first objective value f_2, the best solutions were obtained with the "both" variant (SAT and network), followed by the "SAT" one. We also observe that the fastest executions, in general, were also obtained with "SAT", followed by "both", whereas the other two approaches provided, for some instances, much more difficult LO problems (e.g., for table nine5d, "random" and "network" required 477 and 703 seconds, respectively, while "SAT" and "both" only needed 73.6 and 17.5).

instance	f_3				f_1			
method:	random	SAT	network	both	random	SAT	network	both
australia_ABS	**397**	751	—	995	**52261**	74935	—	87938
bts4	290	235	170	**160**	—	6.315e+09	**4.975e+09**	5.395e+09
cbs	19.82	**19.78**	—	1628	2.7e+06	**2.402e+06**	—	7.506e+09
dale	**0**	**0**	—	—	**494**	**494**	—	—
destatis	3440	**473**	3391	484	**1.33e+09**	1.365e+09	3.303e+09	1.345e+09
five20b	—	**1187**	—	—	—	—	—	—
five20c	2544	—	2105	**1824**	—	—	—	—
hier13	34.44	63.75	**34.29**	55.01	**6.242e+08**	8.608e+08	6.417e+08	7.968e+08
hier13x13x13a	40.58	70.5	**39.02**	62.97	7.85e+08	8.964e+08	**7.11e+08**	8.672e+08
hier13x13x13b	40.58	70.5	**39.02**	62.97	56853	65498	**55199**	62861
hier13x13x13c	40.58	70.5	**39.02**	62.97	482136	555270	**465979**	531976
hier13x13x13d	141	70.5	122	**62.97**	848912	555270	803592	**531976**
hier13x13x13e	142	67.18	84.12	**59.75**	1.061e+07	6.452e+06	7.928e+06	**6.17e+06**
hier13x13x7d	**9.501**	28.84	24.74	26.16	**2.058e+06**	2.651e+06	2.257e+06	2.494e+06
hier13x7x7d	**1.454**	9.001	7.482	11.7	**832149**	1.116e+06	1.109e+06	1.105e+06
hier16	22.12	35.34	**19.91**	26.72	**1.056e+09**	1.318e+09	1.063e+09	1.175e+09
hier16x16x16a	29.46	42.57	**19.34**	42.77	1.178e+09	1.413e+09	**9.378e+08**	1.374e+09
hier16x16x16b	29.46	42.57	**19.34**	42.77	100391	111114	**85627**	109613
hier16x16x16c	29.46	42.57	**19.34**	42.77	847619	941676	**712956**	927035
hier16x16x16d	101	56.8	82.67	**53.51**	2.408e+09	1.61e+09	1.749e+09	**1.511e+09**
hier16x16x16e	101	56.8	82.67	**53.51**	1.653e+07	1.213e+07	1.349e+07	**1.161e+07**
nine12	325	276	267	**201**	2.748e+09	2.718e+09	2.222e+09	**2.181e+09**
nine5d	**260**	336	317	331	**1.693e+09**	2.747e+09	1.753e+09	2.556e+09
ninenew	435	370	**232**	258	1.98e+09	2.64e+09	**1.583e+09**	1.982e+09
osorio	**0**	**0**	**0**	**0**	**15**	**15**	17	17
sbs2008_C	**8088**	159389	15226	159389	**581520**	6.465e+06	1.222e+06	6.465e+06
sbs2008_D_b	**13736**	50510	21322	50323	**1.568e+06**	7.308e+06	2.217e+06	7.127e+06
sbs2008_E	4441	**2991**	6836	**2991**	**272787**	1.097e+06	362380	1.097e+06
table1	**180**	239	373	267	9.244e+13	9.715e+13	1.644e+14	**7.235e+13**
table3	5866	579	922	**436**	3.68e+13	1.651e+13	1.877e+13	**1.512e+13**
table4	5866	579	922	**436**	3.101e+11	1.391e+11	1.582e+11	**1.274e+11**
table5	5866	579	922	**436**	3.096e+08	1.387e+08	1.577e+08	**1.27e+08**
table6	**180**	239	372	266	9.072e+10	9.534e+10	1.613e+11	**7.1e+10**
table7	828	25.18	**24.55**	25.18	1.448e+11	**2.59e+10**	2.606e+10	**2.59e+10**
table8	**0**	**0**	**0**	**0**	**725**	**725**	728	728
targus	0.1973	0.1221	**0.0584**	0.1221	**1.979e+06**	2.023e+06	**1.979e+06**	2.023e+06
toy3dsarah	1.241e+06	**103976**	1.584e+06	104118	**6.825e+14**	1.394e+15	8.757e+14	1.394e+15
two5in6	117	200	**115**	184	**9.857e+08**	1.598e+09	1.045e+09	1.555e+09

— objective value not computed, either by time limit reached or by heuristic failure

Table 8. Values of f_3 and f_1 with the four heuristics for fixing the protection direction of sensitive cells, using the interior-point of CPLEX as solver, order $\pi = (4, 2, 3, 1)$, and allowing a 2% of deviations in cell values. In boldface, the run with the best quality solution.

	MILO-CTA		LO-CTA with CPLEX barrier					
	with CPLEX		Standard		HSD		HSD no-crossover	
instance	CPU	%gap	(4,3,2,1)	(4,2,3,1)	(4,3,2,1)	(4,2,3,1)	(4,3,2,1)	(4,2,3,1)
australia_ABS	1.25	0.00	2.99	3.01	0.6	2.66	**0.56**	2.49
bts4	> 3600	70.31	731.89	3429.09	34.2	> 3600	**30.43**	194.04
cbs	2.27	0.00	0.25	0.37	0.28	0.44	**0.23**	0.34
dale	33.08	0.00	0.52	1.3	0.58	1.33	**0.46**	1.3
destatis	2908.42	0.10	**0.94**	2.66	1.12	4.32	1.05	4.44
five20b	> 3600	100.00	> 3600	> 3600	767.54	> 3600	**689.44**	2973.61
five20c	> 3600	99.99	> 3600	> 3600	2427.46	> 3600	**2203.29**	> 3600
hier13	225	0.07	**3.07**	8.48	3.84	12.17	3.64	11.41
hier13x13x13a	191.59	0.06	**2.79**	25.55	3.37	9.47	3.12	8.1
hier13x13x13b	346.24	0.06	4.09	21.04	3.21	9.68	**3.09**	8.38
hier13x13x13c	462.99	0.01	5.06	20.8	3.44	10.82	**3.12**	9.5
hier13x13x13d	80.16	0.00	4.71	14.81	3.64	11.39	**3.4**	10.34
hier13x13x13e	67.37	0.00	3.67	15.63	3.26	18.42	**3**	11.76
hier13x13x7d	145.69	0.09	0.94	1.83	0.46	1.34	**0.44**	1.24
hier13x7x7d	13.98	0.06	0.22	0.34	0.13	0.26	**0.12**	0.24
hier16	> 3600	43.25	31.27	822.08	19.36	58.66	**18.05**	64.58
hier16x16x16a	> 3600	42.85	**10.2**	114.35	14.48	68.59	13.4	37.13
hier16x16x16b	> 3600	27.34	14.68	112.43	13.67	40.84	**13.21**	38.05
hier16x16x16c	> 3600	31.66	18.45	68.24	15.14	38.92	**13.6**	39.83
hier16x16x16d	> 3600	48.35	**11.4**	95.7	13.67	48.59	14.05	38.78
hier16x16x16e	> 3600	36.47	19.07	61.83	15.17	47.05	**14.76**	39.59
nine12	> 3600	99.94	**29.33**	784.24	46.16	209.8	42.6	197.39
nine5d	> 3600	99.99	239.13	68	31.06	476.65	**27.28**	113.72
ninenew	> 3600	65.67	**21.54**	60.89	27.42	194.9	26.74	73.65
osorio	0.77	0.00	0.3	1.88	0.33	1.74	**0.27**	0.42
sbs2008_C	103.54	0.10	0.31	0.5	0.2	0.53	**0.16**	0.51
sbs2008_D_b	> 3600	5.76	2.31	5.41	1.55	6.26	**1.39**	6.34
sbs2008_E	4.64	0.00	0.08	0.12	**0.06**	0.13	**0.06**	0.12
table1	16.55	0.10	**0.1**	0.24	0.12	0.34	**0.1**	0.33
table3	> 3600	9.27	**1.04**	6.5	1.15	5.11	1.09	5.07
table4	> 3600	8.46	**1.03**	3.77	1.17	5.05	1.11	5.12
table5	> 3600	0.32	**1.04**	3.25	1.17	4.69	1.1	4.71
table6	3.96	0.10	**0.08**	0.19	0.09	0.27	**0.08**	0.25
table7	0.06	0.00	**0.03**	0.16	0.04	0.15	**0.03**	0.13
table8	0.06	0.00	0.04	0.07	**0.03**	0.09	**0.03**	0.08
targus	0.02	0.00	0.02	**0.01**	0.02	0.03	**0.01**	0.02
toy3dsarah	11.96	0.01	**0.15**	0.34	0.16	0.49	**0.15**	0.45
two5in6	> 3600	50.14	**13.82**	145.82	17.34	66.01	16.21	61.9

Table 9. Comparison of runtimes between MILO-CTA and LO-CTA. The CPU time and the optimality gap achieved within the 3600 seconds time limit are given for MILO-CTA. For LO-CTA, CPU times are reported for the two orders ($\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$) and three different CPLEX barrier variants (standard barrier, homogeneous self-dual barrier, and homogeneous self-dual without crossover). The fastest execution is marked in boldface.

6.5 *Comparing LO-CTA to MILO-CTA*

As it was stated in the Introduction, the purpose of LO-CTA is to provide protected tables of similar quality as those obtained with MILO-CTA, with less computational effort. In the comparison between MILO-CTA and LO-CTA we thus focus on these two criteria: efficiency and quality of the solutions.

Table 9 shows the CPU times obtained with CPLEX for MILO-CTA and LO-CTA. For LO-CTA three barrier variants were used: the standard barrier algorithm, corresponding to columns "Standard" in the table; the homogeneous self-dual interior-point (columns "HSD"); and the homogeneous self-dual without crossover (columns "HSD no-crossover"). Each barrier variant was applied to the two orders $\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$. For MILO-CTA the table also reports the optimality gap achieved; large gaps correspond to feasible sub-optimal solutions for instances that exhausted the 3600 seconds time limit. The fastest execution is marked in boldface.

Although MILO-CTA never provided the fastest run, it was very competitive in some instances (such as "australia_ABS", "cbs", "osorio", "table6", "table7" and "table8"). However, in some other cases MILO-CTA required very long executions (it even exhausted the 3600 seconds time limit) whereas LO-CTA computed a solution in one second (e.g., "destatis", "sbs2008_D_b", "table3", "table4" and "table5"). It is clearly observed that, of

	Ratio MILO-CTA/LO-CTA	
instance	(4,3,2,1)	(4,2,3,1)
australia_ABS	0.99	0.83
bts4	1.25	1.05
cbs	0.73	0.69
dale	0.79	0.79
destatis	0.73	0.63
five20b	147.6	103.3
five20c	153.6	101.3
hier13	1.23	0.89
hier13x13x13a	1.08	0.78
hier13x13x13b	1.09	0.78
hier13x13x13c	1.09	0.78
hier13x13x13d	0.89	0.51
hier13x13x13e	0.99	0.47
hier13x13x7d	0.95	0.87
hier13x7x7d	0.85	0.80
hier16	0.95	0.80
hier16x16x16a	0.96	0.75
hier16x16x16b	0.91	0.73
hier16x16x16c	0.91	0.74
hier16x16x16d	0.94	0.63
hier16x16x16e	0.83	0.57
nine12	151.9	119.6
nine5d	251.7	210.3
ninenew	1.06	0.80
osorio	1.00	1.00
sbs2008_C	2.90	2.11
sbs2008_D_b	2.11	1.60
sbs2008_E	1.43	0.89
table1	0.62	0.58
table3	1.07	0.54
table4	1.04	0.52
table5	1.15	0.58
table6	0.61	0.56
table7	1.29	0.18
table8	1.16	1.16
targus	1.00	1.00
toy3dsarah	1.84	1.32
two5in6	1.13	0.94

Table 10. Ratio of distances between original and protected tables, where the numerator is the distance obtained with MILO-CTA and the denominator comes from LO-CTA. A value greater than one means that the table protected with LO-CTA was closer to the original table (thus better) than the table generated by MILO-CTA.

the six LO-CTA combinations tested, the most efficient was the homogeneous self-dual barrier without crossover and with order $\pi = (4, 3, 2, 1)$ (it was the fastest option in 27 out of 38 instances). It is also worth noting that the two homogeneous self-dual columns (with and without crossover) for order $\pi = (4, 3, 2, 1)$ were the only variants able to solve all the instances within the time limit. Indeed, and surprisingly, the homogenous self-dual algorithm outperformed the standard barrier—which is considered the most efficient interior-point algorithm—in most cases. A possible explanation can be found in the extra constraints added to the lexicographic optimization problem (14): they make the problem near-infeasible, and in those situations the homogeneous self-dual method is known to be numerically more stable. It is also seen that in some cases the crossover significantly increased the solution time: for example, for instance "bts4" and order $\pi = (4, 2, 3, 1)$, we reduced from more than 3600 to only 194.04 seconds when the crossover was deactivated. All in all, in general, it can be concluded that LO-CTA outperformed MILO-CTA in terms of running times.

Unlike for the CPU times, there is not a clear and well defined criterion to compare the quality of solutions provided by different protection methods in the field of statistical disclosure control. We thus considered the difference in absolute value, between the original and the protected table, that is, $\sum_{i \in \mathcal{N}} |x_i - a_i|$, which is related to f_1. We proceeded as follows. Firstly, this measure was obtained for both MILO-CTA and LO-CTA with orders $\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$ (using either standard barrier, homogeneous self-dual or homogeneous self-dual without crossover, since they provided the same or very similar solutions). Secondly, the ratio of these measures between MILO-CTA and LO-CTA was

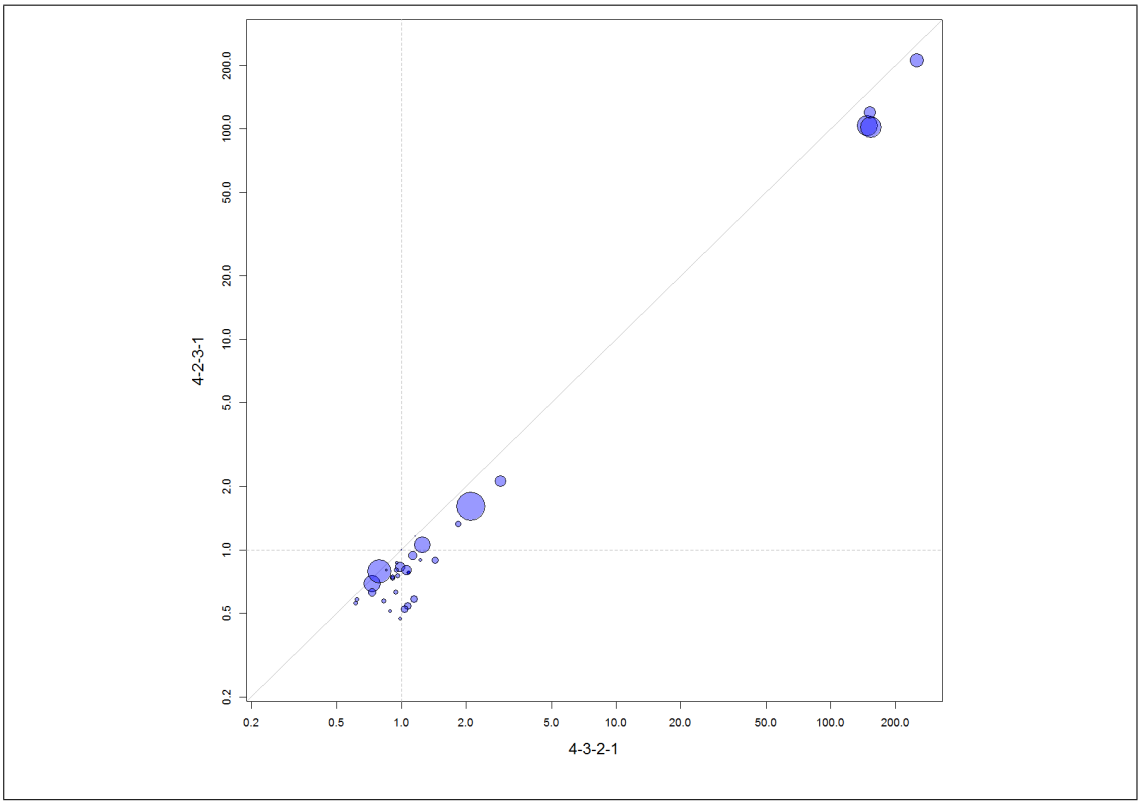

Figure 6.: Plot of ratios of $\sum_{i \in \mathcal{N}} |x_i - a_i|$ between MILO-CTA and LO-CTA with the orders $\pi = (4, 3, 2, 1)$ (horizontal axis) and $\pi = (4, 2, 3, 1)$ (vertical axis). Ratios greater than one mean LO-CTA behaved better than MILO-CTA. Each instance is associated with a circle, whose area is proportional to the number of sensitive cells of the instance.

computed. These ratios are shown in Table 10. Ratios greater than one mean that LO-CTA provided a solution of higher quality than MILO-CTA (i.e., the protected table was closer to the original one). It is seen that LO-CTA clearly ourperformed MILO-CTA in some instances (among them some of the largest ones: "five20b", "five20c", "nine12" and "nine5d"). It could be argued that this is due to the changes in constraints and bounds done by LO-CTA. However this is not the case for two reasons: (1) First, the relative changes made by LO-CTA in constraints and bounds were very small; for example, for "five20b"—one of the largest and most difficult instances—the average relative change in constraints was 0.016, with only four out of 52983 constraints (associated with marginal cell values) with a relative change greater than 1 (the maximum being 2.46). (2) And second, the MILO-CTA approach also reported as optimal some tables that were infeasible (some cells were unprotected, that is, their protection levels were violated); this is due to the big-M constraints of (2) and the coexistence of very small and very large cells in the same table. For example, the number of unprotected cells was 139, 60 and 49 for, respectively, "bts4", "table5" and "sbs2008_D_b" (up to 14—infeasible—tables with unprotected cells were reported as optimal by MILO-CTA).

Figure 6 shows graphically the information in Table 10, in which each circle is associated with an instance, the coordinates of the center of the circle being the two ratios shown in the table. The areas of the circles are proportional to the number of sensitive cells of the table (in theory, the larger the number of sensitive cells, the more difficult should be the MILO-CTA problem). The line $x = y$ of Figure 6 would correspond to instances where the behaviour of LO-CTA is the same for orders $\pi = (4, 3, 2, 1)$ and $\pi = (4, 2, 3, 1)$. The half-plane $x > y$ is associated with instances where the order $\pi = (4, 3, 2, 1)$ provided better solutions than $\pi = (4, 2, 3, 1)$: it is seen that this is the case for all the executions.

This means that $\pi = (4, 3, 2, 1)$ not only provides the fastest executions, but it also gives better solutions than $\pi = (4, 2, 3, 1)$. We also see that a significant number of instances is located in the half-plane $x > 1$, that is, LO-CTA with $\pi = (4, 3, 2, 1)$ outperformed MILO-CTA, and by a large margin in the extreme instances. On the other hand, when MILO-CTA was better (instances in half-plane $x < 1$) the x-coordinates of the points were always in the interval $[0.7, 1]$.

7. Conclusions

The formulation and solution of the new multiobjective LO-CTA data protection problems opens challenges and opportunities. Compared to the original MILO-CTA formulation, it allows to compute a publishable table of similar quality by just solving up to four LO problems. In addition, for different preference orders it is possible to generate different Pareto solutions, each of them with some particular feature. On the other hand, depending on the table it may be needed to tune not only the preference order, but also the algorithm, solver, and allowed percentage of deviations. The resulting problems are challenging for today LO solvers, and they become harder as additional constraints are added during the lexicographic method. Since much larger tables can be easily generated from the current volume of data stored by NSAs, this field of application is a source of huge instances for LO solvers.

Acknowledgments

This work has been supported by MINECO/FEDER grant MTM2015-65362-R, and grant FP7-INFRA-2010-262608 of the European Union. The authors thank the two anonymous reviewers for their insightful comments, which significantly improved the manuscript.

References

[1] M. Bacharach, Matrix rounding problems, Management Science, 9, 732-742, 1966.
[2] B. De Backer, F. Didier, E. Guére, Glop: An open-source linear programming solver, communication at 22nd International Symposium on Mathematical Programming, Pittsburgh, 2015.
[3] D. Baena, J. Castro, J. A. González, Fix-and-relax approaches for controlled tabular adjustment, Computers & Operations Research, 58, 41-52, 2015.
[4] J. Castro, Minimum-distance controlled perturbation methods for large-scale tabular data protection, European Journal of Operational Research, 171, 39–52, 2006.
[5] J. Castro, A shortest-paths heuristic for statistical data protection in positive tables, INFORMS Journal on Computing, 19(4), 520–533, 2007.
[6] J. Castro, Recent advances in optimization techniques for statistical tabular data protection, European Journal of Operational Research, 21, 257–269, 2012.
[7] J. Castro, On assessing the disclosure risk of controlled adjustment methods for statistical tabular data, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 20, 921–941, 2012.
[8] J. Castro, J.A. González, A tool for analyzing and fixing infeasible RCTA instances. Lecture Notes in Computer Science, 6344, 17–28, 2010.
[9] J. Castro, A. Frangioni, C. Gentile, Perspective reformulations of the CTA problem with L_2 distances, Operations Research, 62(4), 891–909, 2014.
[10] J.W. Chinneck, Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods, Springer, 2008.
[11] R.A. Dandekar (2003), Energy Information Administration, Department of Energy, USA. Personal communication.
[12] 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.

[13] P.-P. de Wolf, A. Hundepool, S. Giessing, J.J. Salazar, J. Castro, τ-Argus User's Manual, Statistics Netherlands, 2014. Available online at `http://neon.vb.cbs.nl/casc/Software/TauManualV4.1.pdf`.

[14] J. Forrest, J. Hall, Clp COIN-OR solver, available online at `https://projects.coin-or.org/Clp`.

[15] A. Geoffrion, Proper efficiency and the theory of vector maximization. Journal of Mathematical Analysis and Applications, 22, 618–630, 1968.

[16] S. Giessing, Pre-tabular perturbation with controlled tabular adjustment: some considerations, Lecture Notes in Computer Science, 8744, 48–61, 2014.

[17] J.A. González, J. Castro, A heuristic block coordinate descent approach for controlled tabular adjustment, Computers & Operations Research, 38, 1826–1835, 2011.

[18] M.S. Hernández, J.J. Salazar, Enhanced controlled tabular adjustment, Computers & Operations Research, 43, 61–67, 2014.

[19] A. Hundepool, The Argus software in CENEX, Lecture Notes in Computer Science, 4302, 334–346, 2006.

[20] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Naylor, E. Schulte Nordholt, G. Seri, P.-P. de Wolf, Handook on Statistical Disclosure Control v1.2, Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control, 2010. Available online at `http://neon.vb.cbs.nl/casc/handbook.htm`.

[21] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. Schulte Nordholt, K. Spicer, P.-P. de Wolf, Statistical Disclosure Control. Chichester, Wiley, 2012.

[22] H. Kuhn, A. Tucker, Nonlinear programming. In J. Neyman, editor, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, pages 481–492, 1951. University of California Press, Berkeley, California.

[23] Cs. Mészáros, The BPMPD interior point solver for convex quadratic programming problems, Optimization Methods and Software, 11, 431–449, 1999.

[24] H. Mittelmann, Decision tree for optimization software, available online at `http://plato.asu.edu/guide.html`.

[25] C.M. O'Keefe, P. Gould, T. Churches, Comparison of two remote access systems recently developed and implemented in Australia. Lecture Notes in Computer Science, 8744, 299–311, 2014.

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

[27] K. Soininvaara, T. Oinonen, A. Nissinen, Balancing confidentiality and usability: protecting sensitive data in the case of inward Foreign AffiliaTes Statistics (FATS), Lecture Notes in Computer Science, 8744, 338–349 2014.