Block coordinate descent decomposition for statistical data protection using controlled tabular adjustment

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

Research Report UPC-DEIO DR 2009-10
October 2009

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

Block coordinate descent decomposition for statistical data protection using controlled tabular adjustment

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

Abstract

Tabular data is routinely released by national statistical agencies (NSA) to disseminate aggregated information from some particular microdata. Prior to publication, these tables have to be treated to preserve information without disclosing confidential details from specific respondents. This statistical disclosure control problem is of main interest for any NSA. Most protection techniques rely on the formulation of a large mathematical programming problem, whose solution is computationally expensive even for tables of moderate size. One of these techniques is controlled tabular adjustment (CTA). Although CTA is more efficient than other protection methods, the resulting mixed integer linear problems (MILP) are still challenging. In this work an approach based on block coordinate descent decomposition is designed and applied to large CTA instances. This approach is compared with CPLEX, a state-of-the-art MILP solver. Our results, from both synthetic and real tables with up to 200000 cells, show that the new procedure has a better practical behaviour than a general solver, providing better solutions within a specified time limit (which is required by NSAs in real-world).

Keywords: statistical confidentiality; statistical disclosure control; controlled tabular adjustment; mixed integer linear programming; block coordinate descent; decomposition techniques.

1 Introduction

National statistical agencies (NSAs) routinely disseminate both disaggregated (i.e., microdata or microfiles) and aggregated (i.e., tabular data) information. Tables are generated by crossing two or more categorical variables of a particular microfile (i.e., a census), which results in sets of tables, usually with a large number of cells. NSAs

*Corresponding author

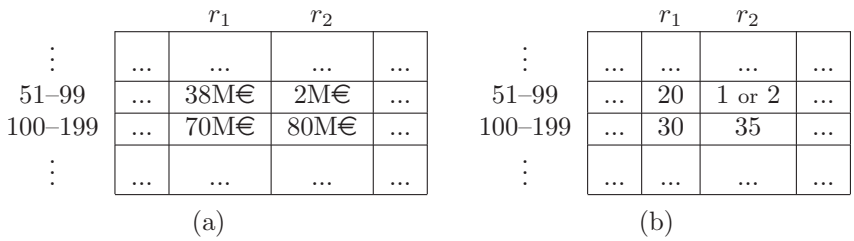

Figure 1: Example of disclosure in tabular data. (a) Net profit per number of employees and region. (b) Number of companies per number of employees and region. If there is only one company with 51-99 employees in region r_2, then any attacker knows the net profit of this company. For two companies, any of them can deduce the other's net profit, becoming an internal attacker.

are obliged by law to guarantee that no particular information from any respondent can be disclosed from the released information. The goal of the statistical disclosure control field is to protect such sensitive information [20]. In this work we focus on tabular data. Some of the state-of-the-art research in this field can be found in the recent monographs [16, 17, 19, 31].

The motivation for this work originated in a project (in collaboration with the NSAs of Germany and The Netherlands) funded by Eurostat, the Statistical Office of the European Communities. The goal of that project was the safe dissemination of European business statistics by Eurostat, which required the protection of tabular data by a technique named controlled tabular adjustment (CTA), to be discussed below. It was observed that, in spite of CTA being more efficient than other tabular protection methods, the solution time required by a general solver to obtain a good or close-to-optimal solution could be large. 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 [12]. This work presents a procedure based on block coordinate descent (BCD) [4, 10] for the solution of CTA formulated as a mixed integer linear problem (MILP). As it will be shown, given a practical time limit (i.e., from some minutes to one hour), for some kinds of tables—namely, hierarchical tables, which are of great practical interest for NSAs—, BCD consistently provided better solutions than a general state-of-the-art solver, such as CPLEX. For general tables, although the benefits of BCD were not so significant, it was also the most efficient approach for the largest instances. Note that approaches similar to BCD, namely branch-and-fix and fix-and-relax, have been used in other large MILPs arising in stochastic programming [1, 15].

Although cell tables show aggregated data for several respondents, there is a risk of disclosing individual information. The example of Fig. 1, from [5], shows a simple case. The left table (a) reports the overall net profit of companies by number of employees (row variable) and region (column variable), while table (b) provides the number of companies. If there were only one company with 51–99 employees in region r_2, then any external attacker would know the net profit of this company. For

Table 1: Result for table "IndustryCode × Size → Var2", from microdata file of τ-Argus distribution.

Method	#supp.	#val. supp.	CPU sec[†]
Hypercube	637	15494253	9
HiTas	528	9016562	15
Shortest-paths	538	8795130	4
Cutting planes	557	7830730	120[*]
Cutting planes	483	7216286	622[*]

[†] Results on a PC with one AMD Athlon 4400+ 64 bits dual core
[*] Time limit

two companies, either one of them could disclose the other's net profit, becoming an internal attacker. These cells which require protection are named sensitive cells. Rules for detecting sensitive cells are beyond the scope of this work (see, e.g., [18] for a recent discussion).

Several techniques are used to control the disclosure of sensitive cells. The most widely used nonperturbative method (i.e., one which does not change cell values) is the cell suppression problem (CSP). The controlled rounding problem (CRP), and the recent controlled tabular adjustment (CTA) are the two more relevant perturbative techniques. An excellent recent survey, mainly focused on CSP and CRP, can be found in [30].

CSP protects the tables by removing some cell contents. It results in a difficult combinatorial optimization problem which finds the optimal pattern (according to some information loss criteria) of additional secondary cells to be removed for protecting sensitive cells (which are removed as well). This problem, initially formulated in [25], has proven to be very difficult, and both exact [22] and heuristic [6, 11] approaches have been developed. The main inconvenience of CSP for NSAs is the impractical execution time required for real and general complex tables. To have a clearer picture, Table 1 shows some results for a toy table obtained with τ-Argus, the state-of-the-art package used by NSAs for the protection of tabular data [24]. This table is obtained from the microdata file accompanying the τ-Argus distribution, crossing categorical variables "industry code" and "size", and using "var2" as explanatory variable. The resulting CSP instance is solved with the four algorithms implemented in τ-Argus: two heuristics which guarantee neither the optimality nor the feasibility— i.e., cells may remain unprotected—(rows "Hypercube" [23] and "HiTas" [14] of Table 1); the feasible heuristic of [6], based on successive shortest-paths (row "Shortest-paths"); and the exact procedure of [22] relying on Benders cutting planes [3] (rows "Cutting planes"), using CPLEX for the subproblems. For this exact approach time limits of 2 and 10 minutes were set. Columns "#supp." and "#val. supp." provide information about the solution reported (number of suppressions, and total value suppressed, respectively). The total value suppressed is the objective function to be minimized. Column "CPU sec" provides the CPU time. The difficulty

Table 2: Sizes of MILPs associated to CSP, CRP and CTA.

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

of this small instance for the exact procedure can be seen. The fast execution of the shortest-paths based heuristic provides an acceptable solution for NSAs, though not optimal; unfortunately this heuristic is not valid for any kind of table [6].

On the other hand, CRP rounds cell values to the closest multiple of a fixed integer rounding base. Table rounding techniques, initially introduced in [2], have been proposed, e.g., in [26, 29]. For many NSAs, the main inconvenience of CRP compared to CTA (discussed below), is that total or subtotal cells, usually associated to national or regional level information, have to be modified to preserve the additivity (thus the feasibility) of the table.

CTA was introduced as an alternative for overcoming some of the drawbacks of previous approaches [5, 13]. The aim of CTA is to find the closest safe and feasible table to the original one, i.e., the minimum distance table satisfying either some upper or lower protection levels for each sensitive cell. It results in a MILP, but much more smaller than those of previous approaches. Table 2 shows the dimensions (constraints, continuous and binary variables) of the CSP, CRP and CTA MILPs, where n is the number of table cells (s being sensitive) and m the number of linear table relations. For instance, note that for a medium-sized table of 4000 cells, 1000 sensitive cells, and 2500 constraints, the size of the CSP/CRP problems is 21 million constraints, and 8 million continuous and 4000 binary variables, respectively. Those problems cannot be dealt with by standard solvers. For this same example, CTA formulates a MILP of 6500 constraints, 8000 continuous and 1000 binary variables, within the limits of state-of-the-art branch-and-cut codes. In spite of the lesser dimension of CTA problems, real-world instances continue to be challenging and may require several hours of execution for an optimal solution (or quasi optimal, e.g., with optimality gaps below 5%). In practice, however, NSAs do not require such optimal solutions, and good, feasible approximate solutions are enough. Indeed, the "true" unknown values of some of the parameters of the CTA model are adjusted by NSAs, and there is therefore no need for an optimal solution. BCD provided good approximate solutions faster than exact procedures for CTA, thus being a promising approach in real applications.

In practice, BCD behaves best when the MILP variables may be clustered, and clusters are loosely coupled. For this reason in this work we mainly focus on hierarchical tables. Hierarchical tables are obtained by crossing groups of categorical variables, and some of them have a hierarchical structure, i.e., some tables are subtables of other tables. Hierarchical tables are of very practical interest for NSAs. The simplest hierarchical table is obtained by crossing two categorical variables, one of them being hierarchical; this particular case is known as 1H2D tables. Figure 2 illustrates a small 1H2D table: rows R_1 and R_3 (related, e.g., to state level information) are decomposed

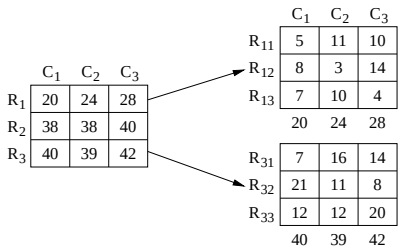

Figure 2: Example of 1H2D table: the row factor in the left table splits into one subtable per row (only two are shown). Rows may be different at each subtable, but columns are the same as in the parent table.

into two subtables (whose rows are, for example, related to region level information); rows of these two subtables could be subsequently decomposed (e.g., for city level information). 1H2D tables can be viewed as a tree of subtables. Note, however, that BCD is not tailored to hierarchical tables. Indeed, in this work it was also applied to general complex tables, though the most successful results were obtained for hierarchical tables. This is partly explained by the inherent block structure of hierarchical tables. Some attempts for extracting the block structure of general tables were made [7], but in general, the resulting blocks were far too coupled for complex instances.

The structure of the paper is as follows. Section 2 outlines the general MILP formulation of CTA. Section 3 introduces the BCD procedure for CTA. Section 4 provides some implementation details and reports the computational results using a set of synthetic and real world tables.

2 Formulation of CTA as a MILP

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

- An array of cells $a_i, i = 1, \ldots, n$, satisfying a set of m linear relations $Aa = b$, $a \in \mathbb{R}^n$ being the vector of a_i's, $b \in \mathbb{R}^m$ the right-hand-side term of the linear relations (usually $b = 0$), and $A \in \mathbb{R}^{m \times n}$ the cell relations matrix. Note that rows of A are made of 1's, 0's and a single -1 (associated to the marginal or total cell value).
- A vector $w \in \mathbb{R}^n$ of positive weights for the deviations of cell values, used in the definition of the objective function.
- A lower and upper bound for each cell $i = 1, \ldots, n$, respectively l_{x_i} and u_{x_i}, which are considered to be known by any data attacker. If no previous knowledge is assumed for cell i then $l_{x_i} = 0$ ($l_{x_i} = -\infty$ if $a \geq 0$ is not required) and $u_{x_i} = +\infty$ can be used.
- A set $\mathcal{P} = \{i_1, i_2, \ldots, i_s\} \subseteq \{1, \ldots, n\}$ of indices of s sensitive or confidential cells.

- A lower and upper protection level for each confidential cell $i \in \mathcal{P}$, respectively lpl_i and upl_i, such that the released values $x \in \mathbb{R}^n$ must 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, according to some distance L, that makes the released table safe. This involves the solution of the following optimization problem:

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

Problem (1) can also be formulated in terms of deviations from the current cell values. Defining

$$z = x - a, \qquad l_z = l_x - a, \qquad u_z = u_x - a, \tag{2}$$

and using the L_1 distance weighted by w, (1) can be recast as:

$$\begin{aligned} \min_{z} \quad & \sum_{i=1}^{n} w_i |z_i| \\ \text{subject to} \quad & Az = 0 \\ & l_z \leq z \leq u_z \\ & z_i \leq -lpl_i \text{ or } z_i \geq upl_i \quad i \in \mathcal{P}, \end{aligned} \tag{3}$$

$z \in \mathbb{R}^n$ being the vector of deviations. Since $w > 0$, introducing variables $z^+, z^- \in \mathbb{R}^n$ so that $z = z^+ - z^-$, the absolute values may be written as $|z| = z^+ + z^-$. Considering a vector of binary variables $y \in \{0,1\}^s$ for the "or" constraints, problem (3) is finally written as a MILP:

$$\min_{z^+, z^-, y} \quad \sum_{i=1}^{n} w_i (z_i^+ + z_i^-) \tag{4a}$$

$$\text{subject to} \quad A(z^+ - z^-) = 0 \tag{4b}$$

$$0 \leq z^+ \leq u_z, \quad 0 \leq z^- \leq -l_z \tag{4c}$$

$$y \in \{0,1\}^s \tag{4d}$$

$$\left.\begin{array}{ll} upl_i\, y_i & \leq z_i^+ \leq u_{z_i} y_i \\ lpl_i (1 - y_i) & \leq z_i^- \leq -lz_i (1 - y_i) \end{array}\right\} i \in \mathcal{P} \tag{4e}$$

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 The block coordinate descent approach for CTA

Block coordinate descent (BCD) solves a sequence of subproblems, each of them optimizing the objective function over a subset of variables while the remaining variables

are kept fixed. This is iteratively repeated until no improvement in the objective function is achieved, e.g., the difference between some (e.g., two) consecutive objective functions is less than a specified optimality tolerance. Convergence of this algorithm is only guaranteed for convex problems where each optimization subproblem has a unique optimizer [4, Prop. 2.7.1] (note that strict convexity satisfies this requirement). Although MILP formulations do not guarantee convergence, BCD usually behaves properly in practical, complex applications [10, 28].

For the particular case of CTA, unless the number of sensitive cells s is small, in general optimal solutions require computationally prohibitive executions. BCD may provide good approximate solutions by optimizing at each iteration the protection sense (either "lower" or "upper") of a subset of sensitive cells, and the deviations for all the cells. The protection senses of the remaining sensitive cells are kept constant to the optimal values of previous iterations. Partitioning the binary variables y of (4a)–(4e) in k blocks, and denoting $y^{j,i}$ as the fixed values of block j at inner iteration i, the algorithm is roughly as follows:

Step 0 Initialization. Set outer iteration counter: $t = 0$. Set initial values, hopefully feasible, to y.

Step 1 $t = t + 1$. Set inner iteration counter $i = 0$. Partition y into k blocks: $y = \{y^{1,i}, \ldots, y^{k,i}\}$, not necessarily of the same size.

Step 1.1 $i := i + 1$. Solve (4a)–(4e) with respect to block $y^{i,i}$, taking into account that $y^{j,i}$ is fixed for $j \neq i$. Let $y^{i,i+1} = (y^{i,i})^*$ (the point at the optimum). Let $y^{j,i+1} = y^{j,i}$ for $j \neq i$.

Step 1.2 If $i < k$ go to Step 1.1.

Step 3 If solution at outer iteration t is better than for $t - 1$, then go to Step 1. Otherwise stop, and return the current optimal solution.

At each major iteration a partitioning of variables y is needed, possibly different: indeed different strategies can be considered between consecutive major iterations, e.g., reversing the order of blocks, changing the blocks, etc. Note that the original problem (4a)–(4e) is solved if only one block of variables is considered. Therefore, although BCD is a heuristic approach, it is easily switched to an optimal approach by setting $k = 1$ at Step 1 for some advanced t. The subproblems of Step 1.1 may be solved by any method; we used the branch-and-cut solver of CPLEX.

Using a true partition of the sensitive cells means that each cell has just one chance of being "upper" or "lower" protected at each major iteration. However, this is not strictly needed, and in practice some sensitive cells could belong to more than one block. In this case the decision variables associated to these cells would be determined more than once for some t.

Several criteria can be used to decide how to divide the cells into the blocks. Two strategies have been tested, which can be viewed as a framework whose implementation would admit many possibilities. The first strategy (named random-BCD) divides $\mathcal{P}$ randomly into a number of blocks, keeping their sizes as similar as possible. The

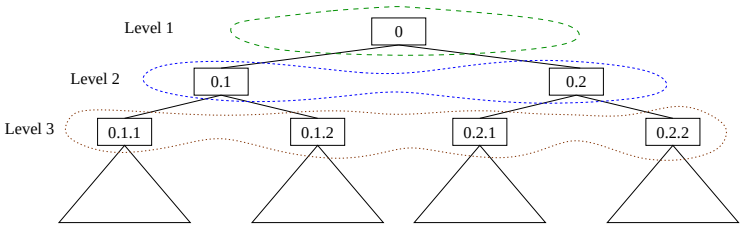

Figure 3: Example of the tree structure of a 1H2D table, variables being partitioned by levels. Boxes refer to the set of sensitive variables in a subtable. Groups of boxes closed by dashed or dotted lines form a block of variables to be optimized together in the first major iteration.

partition is obtained by shuffling the variables such that different blocks are considered at major iterations.

The second strategy (that will be referred to as tree-BCD) is tailored for 1H2D hierarchical tables, exploiting the tree structure of these kinds of tables. In the first major iteration, the sensitive cells are partitioned according to their level: the first group is composed of all the variables of cells in the main table (level 1, or table 0, as seen in Figure 3); then, the second group takes the variables in the next level (0.1, 0.2, and so on); the third group considers all the variables in level 3 (0.1.1, 0.1.2, etc), and so forth until the deepest level. Once the first major iteration is finished, the second one builds overlapping blocks of cells: the first group now includes level 1 and level 2; the second group, level 2 and level 3, etc. A third major iteration makes groups from three consecutive levels. The last major iteration would include all the levels in one group, as a pure CTA problem; if the time limit has not been reached yet, it could benefit from a warm-start from previous solutions.

The previous breadth-first strategy was compared with a depth-first strategy, also implemented. In the latter, blocks were formed with all the subtables from the root to a leaf, taking only one subtable per level. Note that the derived blocks are significantly overlapped. This strategy was not satisfactory, and their results will not be reported in the computational results of Section 4.

One of the main drawbacks of BCD is that dual information for the whole (4a)–(4e) problem is not obtained, and thus the stopping rule only focuses on improvements between consecutive iterations. Note, however, that CTA is a minimum distance problem (1) and that zero is a readily available lower bound. Another main drawback of BCD is that it may not be able to obtain a feasible solution, unless an initial set of feasible either "upper" or "lower" protection senses for y are set at Step 0. For complex and large instances, looking for an initial feasible pattern of protection senses for y is a hard problem (theoretically, as hard as finding the optimal solution). The next subsection describes a heuristic strategy that in practice was very useful.

3.1 Finding a feasible starting point. The SAT method

There are several general approaches for finding an initial point in MILP problems (e.g., variable and node selection, feasibility pump, etc.). Many of them can be found in [9], and are implemented in state-of-the-art solvers. An obvious approach for starting BCD with a feasible point would be to run any of those solvers until the first feasible solution for constraints (4b)–(4e) is found. However, this approach would not exploit the particular structure of CTA. To avoid this lack of exploitation, a particular strategy was developed, which has proven to be successful in most instances. In short, this approach consists of two phases: first, the set of constraints $Az = 0$ is scanned to locate those involving sensitive cells, and each constraint of this subset is analyzed to identify possible infeasible combinations for protection senses y of sensitive cells; and second, all the forbidden combinations are compiled together, and an assignment is sought so that none of these combinations is present. For example, assuming cell values are nonnegative, if one of the original table relations is

$$1_2 + 3_4 + 4_5 + 12_7 = 20_{10},$$

where the subindex denotes the cell index, and the lower and upper protection levels of sensitive cells 4 and 7 are respectively $lpl_4 = upl_4 = 2$ and $lpl_7 = upl_7 = 4$, then $y_4 = y_7 = 1$ (i.e., "upper" protection sense for sensitive cells 4 and 7) is a forbidden solution. Indeed, note that in the protected table this relation would then be

$$x_2 + 5 + x_5 + 16 = 20,$$

which is infeasible since $(x_2, x_5) \geq 0$. The above two phases are outlined below.

The first phase exploits the particular structure of the table relations. Since a table can be described through linear combinations of the cell contents, after rearranging terms, constraint j of $Az = 0$ of (3) can be recast as

$$\sum_{i \in I_j} m_{ij} z_i = \sum_{i \in I'_j} -m_{ij} z_i, \tag{5}$$

where $I_j \subseteq \mathcal{P}$ and $I'_j \subseteq \overline{\mathcal{P}}$ are respectively the sets of sensitive and nonsensitive cells in constraint j, and coefficients m_{ij} are either 1 or -1. Next, lower bounds are computed for each side of (5). The right-hand side is not depending on y, and therefore bounds are

$$\begin{aligned} &\sum_{\substack{i \in I'_j: \\ m_{ij} > 0}} -m_{ij}(u_{x_i} - a_i) + \sum_{\substack{i \in I'_j: \\ m_{ij} < 0}} m_{ij}(a_i - l_{x_i}) \\ &\leq \sum_{i \in I'_j} -m_{ij} z_i \leq \\ &\sum_{\substack{i \in I'_j: \\ m_{ij} > 0}} m_{ij}(a_i - l_{x_i}) + \sum_{\substack{i \in I'_j: \\ m_{ij} < 0}} -m_{ij}(u_{x_i} - a_i). \end{aligned} \tag{6}$$

For some assignment of y, the bounds of the left-hand side are

$$\sum_{\substack{i\in I_j:y_i=0,\\ m_{ij}>0}} -m_{ij}(a_i - l_{x_i}) + \sum_{\substack{i\in I_j:y_i=0,\\ m_{ij}<0}} -m_{ij}lpl_i + \sum_{\substack{i\in I_j:y_i=1,\\ m_{ij}>0}} m_{ij}upl_i + \sum_{\substack{i\in I_j:y_i=1,\\ m_{ij}<0}} m_{ij}(u_{x_i} - a_i)$$

$$\leq \sum_{i\in I_j} m_{ij}z_i \leq$$

$$\sum_{\substack{i\in I_j:y_i=0,\\ m_{ij}>0}} -m_{ij}lpl_i + \sum_{\substack{i\in I_j:y_i=0,\\ m_{ij}<0}} -m_{ij}(a_i - l_{x_i}) + \sum_{\substack{i\in I_j:y_i=1,\\ m_{ij}>0}} m_{ij}(u_{x_i} - a_i) + \sum_{\substack{i\in I_j:y_i=1,\\ m_{ij}<0}} m_{ij}upl_i. \tag{7}$$

The procedure consists of searching for any y so that the bounds of both sides mismatch (i.e., either the lower bound of the left-hand side is greater than the upper bound of the right-hand side, or the upper bound of the left-hand side is less than the lower bound of the right-hand side). At present, mismatching between (6) and (7) is exhaustively searched. Although the matrix A is very sparse and there are usually few sensitive variables at each constraint, the number of combinations to check is $2^{|I_j|}$, so the search is prohibitive even for moderate $|I_j|$. Some procedure to drastically reduce the effective number of combinations to be explored is a matter of further research. In the current version a limit (e.g., 20) is set on the number of sensitive cells to consider, although this means that some infeasible combinations are not detected.

In the second phase all the infeasible combinations for y detected in the first phase are collected, and a solution that avoids all of them is looked for. Note that avoiding all these infeasible combinations is a necessary but not a sufficient condition for an initial feasible point. However, in practice, the resulting solution was generally feasible. This problem is known as the Boolean Satisfiability (SAT), and it consists of determining a satisfying variable assignment for a Boolean function, or determining that no such assignment exists. The subject of practical SAT solvers has received considerable research attention, with many algorithms proposed and implemented, e.g. GRASP [27] and SATO [32]. The solver used in this work was MiniSAT [21], an open-source implementation of SAT techniques.

In general, a solver operates on problems specified in conjunctive normal form. This form consists of the logical "and" of one or more clauses (related to one infeasible combination), which themselves consist of the logical "or" of one or more literals (related to y variables). For instance, suppose that three combinations have been detected as infeasible:

$$\begin{array}{lll} (1) & y_1 = 1, y_2 = 0, y_3 = 1, y_4 = 1 & \Rightarrow y_1 \wedge \neg y_2 \wedge y_3 \wedge y_4 \\ (2) & y_3 = 1, y_2 = 1, y_4 = 1 & \Rightarrow y_3 \wedge y_2 \wedge y_4 \\ (3) & y_5 = 1, y_2 = 0, y_1 = 1 & \Rightarrow y_5 \wedge \neg y_2 \wedge y_1. \end{array}$$

None of these combinations is desired, so we seek for an assignment of y in such a way that all of them are false. Equivalently, we can negate them to find an assignment satisfying (as true) all clauses:

$$(\neg y_1 \vee y_2 \vee \neg y_3 \vee \neg y_4) \wedge (\neg y_3 \vee \neg y_2 \vee \neg y_4) \wedge (\neg y_5 \vee y_2 \vee \neg y_1).$$

This expression can be satisfied with $(\neg y_1 \wedge \neg y_3)$, or numerically $y_1 = 0$ and $y_3 = 0$. This condition is sufficient for the SAT problem, so the other variables can take any value.

Once a solution of the corresponding SAT problem is available, the problem (4a)-(4e), with y fixed, can be checked to determine if there exists a feasible solution for z^+ and z^- . Though the assignment obtained by y does not guarantee the feasibility of the constraints without sensitive variables, it was usually found that SAT returned a feasible point, so that BCD was ready to start. In those few cases where the SAT assignment failed, other starting y values were found by a MILP solver, stopping at the first feasible point. The unsuccessful SAT values of y were used as a warm-start for the MILP solver.

4 Computational results

4.1 Implementation

The BCD approach (including the SAT heuristic for the binary initial point) was implemented in C++. The code allows the user to switch between random-BCD, tree-BCD (only breadth-first strategy, the most efficient option), and the solution of CTA by branch-and-cut (BCD subproblems are solved with this same branch-and-cut). The optimizer used was CPLEX version 11.

The BCD variants may be tuned with some parameters chosen by the user. The most significant parameters are the total time limit (also applicable to the branch-and-cut option); the time limit for each subproblem; the optimality gap for each subproblem; and the number of blocks (subproblems) to be considered. Note that a time limit is required by NSAs for data publication deadlines in the real-world. The random-BCD variant is also affected by the randomness of the blocks selection. After exploring the effect of these factors with several instances, it can be concluded that BCD is barely sensitive both to random variability and the user adjustments of parameters. No significant differences were found if the optimality gap of subproblems is set to values between 1% to 10% (with lower gaps, subproblems may achieve better solutions, but they take more time, so fewer subproblems can be solved due to the overall time limit). For the total time limit, the larger it is, the better the solution found; for the subproblem time limit, it should be large enough to improve the initial provided solution, but not too large to avoid that most of the total time is spent on too few subproblems.

With regard to the number of blocks in the random strategy, it was observed that it is inadvisable to take a large number, since each subproblem would consider too few sensitive variables at once, which is unlikely to improve the previous solution. In our tests, values from 2 to 40 blocks were chosen, noticing that low numbers (say, below 10) are preferable, with no significant differences between them. However, in general it is recommended to take more than three or four blocks —or even more, depending on the table size— since lower numbers lead to large subproblems that could take as much time as the original problem.

Table 3: Characteristics of tables

instance	n	$\lvert\mathcal{P}\rvert$	m	N. coef.
table 11	20280	973	1560	41314
table 12	21476	2062	1684	43784
table 13	36806	3345	5832	76087
table 14	5388	224	1474	11346
table 15	26884	2443	3368	54681
table 16	52063	4732	6900	106282
table 17	16852	1531	2390	34551
table 18	8316	755	1339	17204
table 21	126362	12324	5501	255102
table 22	43365	4128	3577	88221
table 23	71640	10442	3430	144684
table 24	166248	12927	7966	335808
table 25	55620	4323	2877	112536
table 26	209456	16110	12164	422994
table 27	65241	4744	7240	131780
table 28	88164	6854	4321	178164
hier13x13x13d	2197	108	3549	11661
hier16	3564	224	5484	19996
ninenew	6546	858	7340	32920
nine12	10399	1178	11362	52624

4.2 Test instances

The BCD approach was tested using both 1H2D (hierarchical) and general complex tables. Real-world tables are provided by NSAs as a set of equations, without providing information about the particular inherent structure (which is difficult to be extracted by general procedures [7]). Hierarchical tables were thus obtained by a generator of 1H2D synthetic tables. Some of the parameters of the generator controlling the dimensions of the table are: mean number of rows in a table; number of columns per subtable; depth of hierarchical tree; minimum and maximum number of rows with hierarchies for each table; and probability for a cell to be marked as sensitive. The random generator is available from `http://www-eio.upc.es/~jcastro/generators_csp.html`. A set of 16 representative and large 1H2D tables was generated. Their main dimensions are reported in Table 3. Columns n, $|\mathcal{P}|$, m and "N. coef." show, respectively, the number of cells, sensitive cells, table linear relations (i.e., rows of matrix A), and nonzero coefficient of matrix A. The last eight 1H2D tables are somewhat larger than the first eight. The depth of the hierarchical tree of each instance is six, but tables 16 and 17 (five) and table 18 (four).

For the general tables, four of the most complex instances of CSPLIB (a library of tabular data protection instances [22], available from `http://webpages.ull.es/users/casc/#CSPlib:`) were considered. Their dimensions are also reported in Table 3.

4.3 Results

The two BCD variants, random-BCD and tree-BCD, and the state-of-the-art branch-and-cut of CPLEX were compared using the previous set of instances. All the executions were carried out on a Linux Dell PowerEdge 6950 server with four dual core AMD Opteron 8222 3.0 GHZ processors (without exploitation of parallelism capabilities) and 64 GB of RAM. For the 1H2D tables, the sequence of feasible solutions obtained for each method was recorded, until the time limit was reached. Our goal was to show that, in the case of early interruption of the optimization process, the quality of the best solution reached using BCD was similar or better than the best solution provided by CPLEX branch-and-cut. The evolution of the feasible solutions obtained is shown in Figures 4 for the smaller instances, and 5 for the larger ones. In those figures, lines "BCD" refer to random-BCD, "Tree" to tree-BCD and "B& Cut" to branch-and-cut solutions.

For the instances of Figure 4, the total time limit was set to two hours. For the two BCD variants, the subproblem time limit was one hour. In all cases but the smallest table 14, branch-and-cut exhausted the time limit without reaching an optimality gap of 5% (table 14 took less than 15 minutes to be solved, but it is significantly smaller than the rest). In some cases the total running time was somewhat superior to two hours (as in table 16), because tree-BCD subproblems exhausted the subproblem time limit, thus exceeding the total time of two hours. This is because tree-BCD solves a short sequence of subproblems with increasing difficulty. In the first major iterations an optimal solution to the subproblems may quickly be found; in the last major iterations, the subproblems may take a considerable amount of time. Although the procedure may eventually attempt to solve the whole CTA problem, this never happened within the time limit set for the performed tests. On the other hand, random-BCD usually consists of a longer sequence of subproblems, which are solved more quickly, though new solutions often do not improve on the previous. For the instances of Figure 4, five blocks of cells were considered for random-BCD.

In the plots of Figure 4, it is seen that the objective function reaches an almost steady state relatively soon. Little improvement is observed, in general, in the last part of the optimization process. Indeed, the larger tables (except for number 14 and number 18) could not be solved within an optimality tolerance of 5% even for much longer time limits. For instance, tables 13 and 16 were tried with a time limit of two days of CPU time. The best objective function value for table 13 was 453459, with an optimality gap of 45%; for table 16 the best objective function value was 608223, with a gap of 55%. Although both solutions are slightly better than those reached by BCD, it is not clear that the improvement in the objective function is worth the computational effort.

Another noticeable conclusion from Figure 4 is that, in general, BCD variants were faster in achieving and acceptable solution. For these test tables the SAT technique returned feasible points in all cases, and they were of better quality than the first feasible points provided by branch-and-cut. CPLEX branch-and-cut started from very poor quality feasible solutions, though they were quickly improved upon. However, in the long run, it is likely that branch-and-cut improves any BCD method. Finally, it is also observed that random-BCD and tree-BCD are very similar, the former being

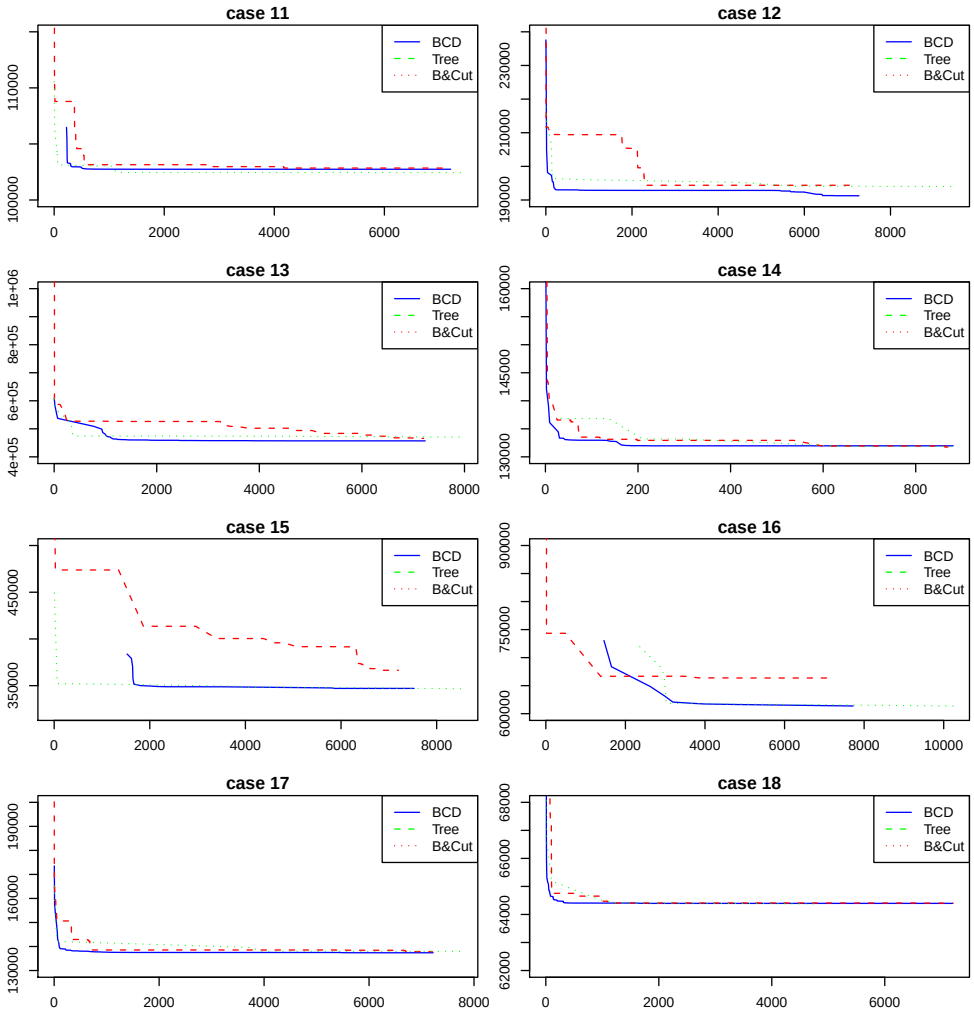

Figure 4: Sequences of feasible solutions for tables 11 to 18. The horizontal axis shows the CPU time and the vertical shows the best objective function value achieved. The time limit was two hours except for table 14, since branch-and-cut found a solution in less than 15 minutes.

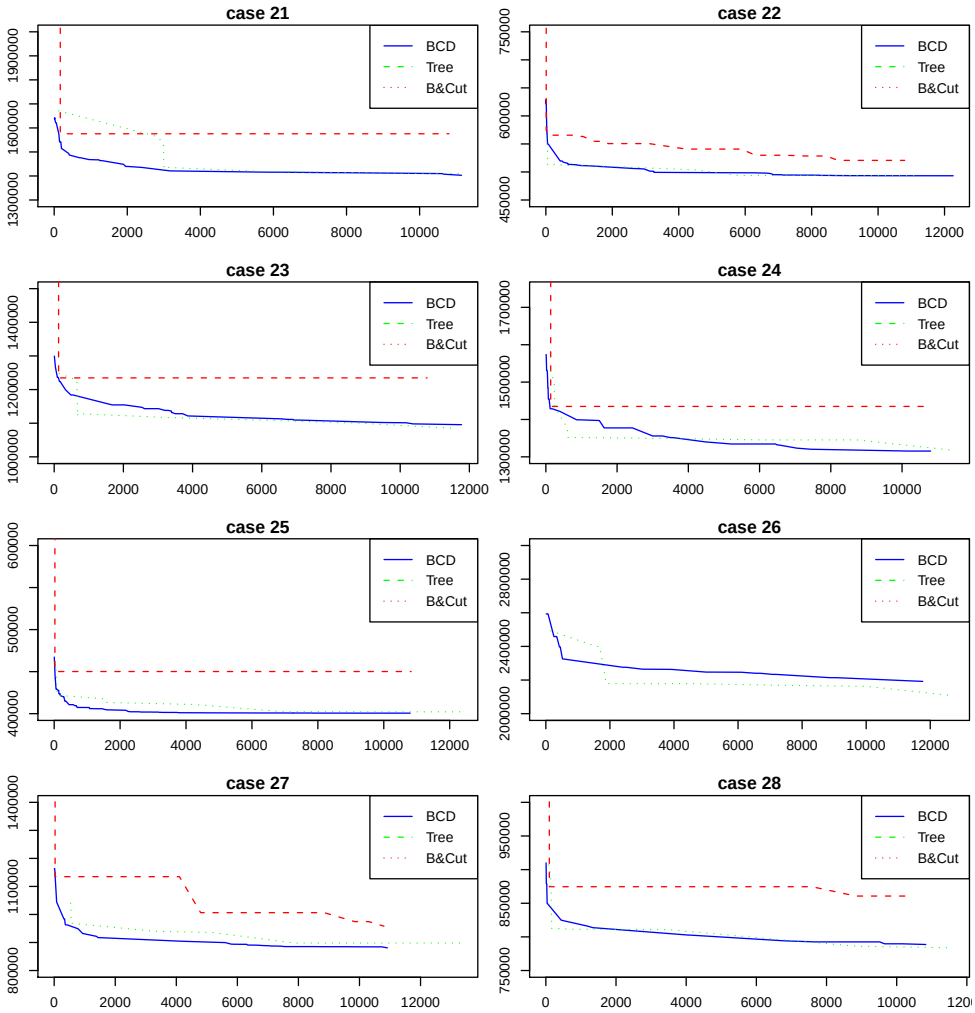

Figure 5: Sequences of feasible solutions for tables 21 to 28. The horizontal axis shows the CPU time and the vertical axis the best objective function value achieved. The time limit was three hours. The branch-and-cut solution for tables 26 is not shown because the best value of the objective function was $5.33 \cdot 10^9$.

Table 4: Results for general tables

instance	B&C	BCD	Time limit
hier13x13x13d	416604	477024	350 sec.
hier16	$5.47 \cdot 10^8$	$6.94 \cdot 10^8$	7200 sec.
ninenew	$9.35 \cdot 10^8$	$5.90 \cdot 10^8$	7200 sec.
nine12	$1.41 \cdot 10^9$	$9.32 \cdot 10^8$	7200 sec.

slightly better. This is a surprising conclusion, since tree-BCD was supposed to take advantage of the tree structure of the tables.

Figure 5 shows the output obtained for the second group of larger tables. For these instances the total time was three hours, and ten blocks were used for random-BCD. The previous remarks for the results of Figure 4 are also valid for this set of tables. The advantage of BCD can even be emphasized with respect to a standard branch-and-cut approach: the latter often gets stuck (at least, in the three-hour time limit considered), whereas the objective of BCD sequences tends to decrease. The branch-and-cut sequence for table 26 is not shown in Figure 5, since the time limit was reached with a (bad) solution of objective equal to $5.33 \cdot 10^9$.

Table 4 shows the results obtained for the four complex general tables of Table 3. Columns "B&C" and BCD report the objective function value reached within the CPU time limit indicated in the column "Time limit" for, respectively, the CPLEX branch-and-cut and random-BCD. From these results, it cannot be concluded that BCD is in general competitive for general complex tables against a state-of-the-art branch-and-cut. This is partly explained by the complexity of the cell tables' interrelations, which result in highly coupled clusters of cells in the BCD approach. However, in the significantly two largest instances (ninenew and nine12) BCD returned a better solution within the time limit.

5 Conclusions

From our actual experience with real-world instances, it can be stated that CTA problems can be extremely difficult for large and complex tables, even for state-of-the-art branch-and-cut solvers. The BCD approach presented and tested in this work was able to obtain good solutions within one or two hours of the CPU time limit, for large (up to 200000 cells, 16000 being sensitive) 1H2D tables. Moreover, the solution obtained was comparable, and usually better than the incumbent provided by state-of-the-art branch-and-cut solvers within the same time limit. For general tables (with unknown internal structure), BCD did not outperform branch-and-cut for all the tables tested, but it did for the largest tables. We are thus optimistic about the possibilities of the method for even more difficult real-world tables (with a higher number of cells and sensitive cells). This is partly supported by the better observed behaviour of random-BCD against tree-BCD: random-BCD can be immediately applied to more general (other than 1H2D) classes of tables, without need to exploit the particular internal structure of the table relations.

The (increasing) ability of NSAs to create more complex and huge tables from collected data is an incentive to develop powerful tools for CTA. Among them we find Benders' reformulation of CTA; some preliminary testing with a prototype showed the approach is efficient for two-dimensional tables [8], but deeper cuts are needed for more complex tables. Other data protection approaches, like interval protection, which results in a massive linear programming problem, and its efficient solution by structured interior-point methods, are also among the remaining tasks to be addressed in this challenging field.

6 Acknowledgments

The authors thank Daniel Baena (from the NSA of Catalonia) for generating the tables used in the computational results. This work has been supported by the Spanish MEC grant MTM2006-05550.

References

[1] Alonso-Ayuso, A., Escudero, L.F., and Ortuño, M.T. (2003), BFC, A branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs, *European Journal of Operational Research*, 151, 503–519.

[2] Bacharach, M. (1966), Matrix rounding problems, *Management Science*, 9, 732–742.

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

[4] Bertsekas, D.P. (1999), *Nonlinear Programming, 2nd ed.*, Athena Scientific, Belmont.

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

[6] Castro, J. (2007), A shortest paths heuristic for statistical disclosure control in positive tables, *INFORMS Journal on Computing* 19, 520–533.

[7] Castro, J., and Baena, D. (2006), Automatic structure detection in constraints of tabular data, *Lecture Notes in Computer Science*, 4302, 12–24.

[8] Castro, J., and Baena, D. (2008), Using a Mathematical Programming Modeling Language for Optimal CTA, *Lecture Notes in Computer Science*, 5262, 1–12.

[9] Chinneck, J.W. (2008), *Feasibility and Infeasibility in Optimization*, Springer, New York.

[10] Conejo, A.J., Castillo, E., Minguez, R., and Garcia-Bertrand, R. (2006), *Decomposition Techniques in Mathematical Programming: Engineering and Science Applications*, Springer, Berlin.

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

[12] Dandekar, R.A. (2003) (Energy Information Administration, Department of Energy, USA.) Personal communication.

[13] 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`).

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

[15] Dillenberger, Ch., Escudero, L.F., Wollensak, A., and Zhang, W. (1994), On practical resource allocation for production planning and scheduling with period overlapping setups, *European Journal of Operational Research*, 75, 275–286.

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

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

[18] Domingo-Ferrer, J., and Torra, V. (2002), A critique of the sensitivity rules usually employed for statistical table protection, *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 10, 545–556.

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

[20] Domingo-Ferrer, J., and Torra, V. (2004), Disclosure risk assessment in statistical data protection, *Journal of Computational and Applied Mathematics* 164–165, 285–293.

[21] Eén, N., and Sörensson, N. (2003), An extensible sat-solver, in *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing.*

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

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

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

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

[26] Kelly, J.P., Golden, B.L., Assad, A.A., and Baker, E.K. (1990), Controlled rounding of tabular data, *Operations Research*, 38, 760–772.

[27] Marques-Silva, J.P., and Sakallah, K.A. (1999), GRASP: A search algorithm for propositional satisfiability, *IEEE Transactions on Computers* 48, 506–521.

[28] Plazas, M.A. (2006), *Multistage stochastic model for bidding in electrical markets* (in Spanish), Ph.D. Thesis, Universidad de Castilla-La Mancha.

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

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

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

[32] Zhang, H. (1997), SATO: An efficient propositional prover, in *Proceedings of the International Conference on Automated Deduction*, 272–275, July 1997.