A heuristic block coordinate descent approach for 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 2010-06
September 2010; updated February 2011

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

A heuristic block coordinate descent approach for 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

One of the main concerns of national statistical agencies (NSAs) is to publish tabular data. NSAs have to guarantee that no private information from specific respondents can be disclosed from the released tables. The purpose of the statistical disclosure control field is to avoid such a leak of private information. Most protection techniques for tabular data rely on the formulation of a large mathematical programming problem, whose solution is computationally expensive even for tables of moderate size. One of the emerging techniques in this field 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 a heuristic approach based on block coordinate descent decomposition is designed and applied to large hierarchical and general 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 1,200,000 cells, 100,000 of them being sensitive (resulting in MILP instances of up to 2,400,000 continuous variables, 100,000 binary variables, and 475,000 constraints) show that the heuristic block coordinate descent has a better practical behaviour than a state-of-the-art solver: for large hierarchical instances it provides significantly better solutions within a specified realistic time limit, as required by NSAs in real-world.

Keywords: statistical confidentiality; statistical disclosure control; controlled tabular adjustment; mixed integer linear programming; heuristics; 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

[*] Corresponding author

		r_1	r_2	
⋮	...	...	...	...
51–99	...	380M€	35M€	...
100–199	...	700M€	800M€	...
⋮	...	...	...	...

(a)

		r_1	r_2	
⋮	...	...	...	...
51–99	...	20	1 or 2	...
100–199	...	30	35	...
⋮	...	...	...	...

(b)

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.

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 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 [18]. 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 [13, 14, 15, 17, 27].

Although cell tables show aggregated data for several respondents, there is a risk of disclosing individual information. The example of Figure 1 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 two companies, either one of them could disclose the other's net profit, becoming an internal attacker. Even if the number of companies in this cell was larger, but one of them could obtain a good estimator of another's net profit (for instance, by subtracting its own contribution from the cell value), the cell would be considered unsafe. These unsafe cells which require protection are named sensitive cells. Rules for detecting sensitive cells are beyond the scope of this work (see, e.g., [16, 23] for a recent discussion).

There are several available techniques for the protection of tabular data. The most widely used nonperturbative method (i.e., one which does not change cell values) is named cell suppression problem (CSP) [5, 24]. CSP has been and is one of the preferred options for NSAs. Recently, a new perturbative approach (i.e., cell values are slightly adjusted or modified) named controlled tabular adjustment (CTA) was introduced [4, 11]. Although it is a recent approach, CTA is gaining recognition among NSAs. For instance, CTA appears in [23] among the list of currently available techniques for tabular data, and it is considered as one of the emerging methods. In addition, we recently developed in collaboration with the NSAs of Germany and the Netherlands [21] a package for CTA in the scope of two projects funded by Eurostat, the Statistical Office of the European Communities. The goal of those projects was the

Adjusted table,
lower protection direction

		r_1	r_2	
⋮	...	...	...	...
51–99	...	385M€	30M€	...
100–199	...	695M€	805M€	...
⋮	...	...	...	...

Adjusted table,
upper protection direction

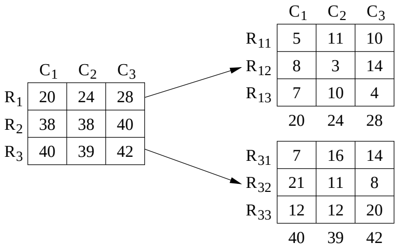
		r_1	r_2	
⋮	...	...	...	...
51–99	...	375M€	40M€	...
100–199	...	705M€	795M€	...
⋮	...	...	...	...

Figure 2: Protected (or adjusted) tables with “lower protection direction” and “upper protection direction” considering lower and upper protection levels of five (i.e., value of sensitive cell is respectively reduced and increased by five units) for the original table of Figure 1.

safe dissemination of European business and animal production statistics by Eurostat, and CTA was used within the overall protection scheme. CTA is thus a method of actual practical relevance. It is worth noting that in recent specialized workshops on statistical disclosure control, relevant members of NSAs stated that perturbative methods, like CTA, are gaining acceptance [28], and some perturbative approaches are being used for the protection of national census tables (e.g., for Germany [20]).

Figure 2 illustrates CTA on the small two-dimensional table of Figure 1, considering cell (51–99,r_2) as the only sensitive cell, with lower and upper protection levels of 5. Note that protection levels are either directly obtained from the rules that provide the set of sensitive cells, or just computed as a percentage of the cell value. Depending on the “protection direction” of the sensitive cell, either “downward” or “upward”, which has to be decided, the value to be published for this cell will be respectively less or equal than the original cell value minus the lower protection level, or greater or equal than the original cell value plus the upper protection level. In the example of Figure 2, if the protection direction is “down”, then the value published for the sensitive cell should be less or equal than 30; a possible adjusted table for this case is shown in the left table of Figure 2. If the protection direction is “up”, then the value must be greater or equal than 40, as shown in the right table of Figure 2. Note that the remaining cells have to be adjusted to preserve the value of marginal cells. CTA aims at finding the protection direction of each sensitive cell (binary decisions), and the adjustments to be done to each cell (continuous variables), preserving the value of total cells (or some subset of particular cells, possibly empty), and minimizing the distance between the original and the released table.

Although CTA formulates mixed integer linear problems (MILPs) with a number of variables and constraints that is linear in the size of the table —and thus its solution can be attempted with generic state-of-the-art solvers—, real-world instances are challenging and may require many hours of execution for an optimal solution (or quasi optimal solution, 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, a big effort to reach an optimal solution may not make sense, since some of the parameters of the CTA model are estimated by NSAs (as, e.g., the amount



	C_1	C_2	C_3
R_1	20	24	28
R_2	38	38	40
R_3	40	39	42

	C_1	C_2	C_3
R_{11}	5	11	10
R_{12}	8	3	14
R_{13}	7	10	4
	20	24	28

R_{31}	7	16	14
R_{32}	21	11	8
R_{33}	12	12	20
	40	39	42

Figure 3: Example of 1H2D table: the row factor in the left table splits into one subtable per row (only two are shown). The number of rows may be different at each subtable, but the number of columns are the same as in the parent table.

of protection to be provided to a certain table). Moreover, 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 [10]. This work presents a heuristic procedure based on block coordinate descent (BCD) [3, 9] for the solution of CTA formulated as a 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, discussed below, 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, real-world, instances.

As far as we know, this is the first heuristic approach for real-world large CTA instances. A previous work [22] applied some metaheuristics and learning heuristics only to small tables (two-dimensional tables of up to 625 cells), while we consider more complex hierarchical tables of about 1,200,000 cells (100,000 of them sensitive). We also tried other general metaheuristics as genetic algorithms without success. Indeed, combinations or slight modifications of solutions are not expected to satisfy the large number of linear equations with no particular structure of CTA. Therefore, for the purpose of guaranteeing this large number of linear constraints, any practical heuristic should rely on MILP technology, as BCD does.

Briefly, BCD is a simple strategy that decomposes the problems in blocks (or clusters) of variables, and sequentially solves the MILP subproblem for each block. Therefore, 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 a number 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 3 illustrates a small 1H2D table: rows R_1 and R_3 (related, e.g., to state level information) are decomposed 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 [6] but, in general, the resulting blocks were far too coupled for complex instances. Note that, in practice, tables (either 1H2D or general) are represented by a set of linear equations imposing each of them that the sum of some set of cells equals a given marginal cell. This model will be used in the CTA formulation of next section.

There are other approaches similar to BCD, namely fix-and-relax and branch-and-fix, which have been used in other large MILPs arising, respectively, in scheduling and stochastic programming problems [1, 12]. As BCD, fix-and-relax decomposes the problem in k blocks of binary variables. It iteratively optimizes for each block i of binary variables, fixing the variables of blocks $j < i$, and relaxing the variables of blocks $j > i$, unlike BCD, which also fixes the variables of blocks $j > i$. Moreover, fix-and-relax performs one single iteration, whereas BCD keeps on iterating while the solution is improved. Branch-and-fix is a significantly different procedure, but it has in common with the other approaches the idea of decomposing a large MILP in smaller subproblems, optimizing —in a coordinated way— all of them by fixing some common binary variables. Full details can be found in [1].

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, and describes an approach based on the Boolean Satisfiability problem (SAT for short, hereafter) for obtaining good initial feasible solutions. 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 $\mathcal{S} = \{i_1, i_2, \ldots, i_s\} \subseteq \{1, \ldots, n\}$ of indices of s sensitive or confidential cells.
- A lower and upper protection level for each confidential cell $i \in \mathcal{S}$, 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:

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

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

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

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

$$\begin{array}{rl} \min\limits_{z} & \sum\limits_{i=1}^{n} w_i |z_i| \\ \text{subject to} & Az = 0 \\ & l_z \leq z \leq u_z \\ & z_i \leq -lpl_i \text{ or } z_i \geq upl_i \quad i \in \mathcal{S}, \end{array} \tag{3}$$

$z \in \mathbb{R}^n$ 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 an MILP:

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

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

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

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

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

When $y_i = 1$ the constraints mean $upl_i \leq z_i^+ \leq u_{z_i}$ and $z_i^- = 0$, thus the protection direction is "upward"; when $y_i = 0$ we get $z_i^+ = 0$ and $lpl_i \leq z_i^- \leq -l_{z_i}$, thus the protection direction is "downward".

3 The heuristic 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 (for instance, 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 [3, Prop. 2.7.1] (note that strict convexity satisfies this requirement). Although MILP problems are not convex, and thus they do not guarantee convergence, BCD usually behaves properly in practical complex applications [9, 26], and it can be used as a heuristic approach.

For the particular case of CTA, if the number of sensitive cells s is very large, optimal solutions may require computationally prohibitive executions. BCD may provide good approximate solutions by optimizing at each iteration the protection direction (either "downward" or "upward") of a subset of sensitive cells, and the deviations for all the cells. The protection directions of the remaining sensitive cells are kept constant at the optimal values of previous iterations. Note that continuous variables of the problem (the deviations for all the cells) are never fixed; blocking and fixing is only performed for the binary variables, which is a significant difference with the standard BCD method. Partitioning the binary variables y of (4a)–(4e) into 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 $i = 0$.
Divide y into k blocks: $y = \{y^{1,i}, \dots, y^{k,i}\}$, 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 2 Check for end conditions: if apply, stop, and return the current best solution. Otherwise, go to Step 1

Note that the original problem (4a)–(4e) is solved if only one block of variables is considered. Therefore, although BCD is a heuristic approach for MILP problems, 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 MILP method; we used the branch-and-cut solver of CPLEX.

Different strategies can be considered for the division of variables y between consecutive major iterations, e.g., reversing the order of blocks, changing the blocks, etc.

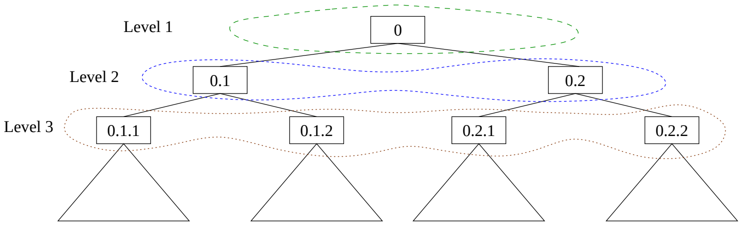


Figure 4: Example of the tree structure of a 1H2D table, variables being partitioned by levels. Boxes refer to the set of sensitive variables in a subtable. Groups of boxes closed by dashed or dotted lines form a block of variables to be optimized together in the first major iteration.

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.

Two strategies have been tested, which can be viewed as a framework whose implementation would admit further possibilities. The first strategy (named random-BCD) divides $\mathcal{S}$ randomly into a number of blocks, keeping their sizes as similar as possible. The 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 block is composed of all the variables of cells in the main table (level 1, or table 0, as seen in Figure 4); then, the second block takes the variables in the next level (0.1, 0.2, and so on); the third block 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 block now includes level 1 and level 2; the second one, level 2 and level 3, etc. A third major iteration makes blocks from three consecutive levels. The last major iteration would include all the levels in one block, 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 are not 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 “up” or “down” protection directions for y are set at Step 0. For complex and large instances, looking for an initial feasible pattern of protection directions 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. Indeed, as it will be noted in the computational results of Section 4, this heuristic for CTA instances provided better initial solutions than the several strategies tested with CPLEX.

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 [8], 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. Briefly, 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 directions 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., protection direction for sensitive cells 4 and 7 is “up”) 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 $I_j \subseteq \mathcal{S}$ and $I'_j \subseteq \overline{\mathcal{S}}$ are respectively the sets of sensitive and nonsensitive cells in constraint j, and coefficients m_{ij} are elements of the matrix A, 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

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

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

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

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).

Provided that the number of sensitive variables at each constraint, $|I_j|$, is small (say, less than 20), exhaustive search could be used to find every infeasible combination. However, the cost is prohibitive even for moderate $|I_j|$, so we have developed a simple but very efficient method to detect the infeasible combinations in a constraint. At first, the procedure determines which combination of y is giving the highest value of the lower bound of (7). If this value is greater than the upper bound of (6), one infeasible combination has been found, and others may be around. Therefore, starting from the first one, we go through each combination that arises by changing only one direction and that it has not been visited before. The combination is pruned, and hence discarded, if the resulting lower bound of (7) falls below the upper bound of (6); otherwise it is included in a list of combinations to be explored, until all of them have been pruned. The same process is repeated for combinations under the lower bound of (6). Usually only a few combinations appear at each constraint, so the search is very efficient, whatever the number of sensitive cells in it.

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 [25] and SATO [29]. The solver used in this work was MiniSAT [19], 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:

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

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. Note that BCD subproblems are solved with this same branch-and-cut for a fair comparison. The SAT heuristic may be activated to provide an initial point for the branch-and-cut method as well, such that the benefits of either the SAT heuristic, or the BCD approach, or both of them together, can be isolated. 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 smaller 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.

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 [6]). 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 24 representative and large 1H2D tables was generated. Their main dimensions are reported in Table 1. Columns n, s, m and "N. coef." show, respectively, the number of cells, sensitive cells, table linear relations (i.e., rows of matrix A), and nonzero coefficients of matrix A. Columns "cont.", "bin." and "constr." show the number of continuous variables, binary variables, and constraints of the resulting MILP problem (4a)–(4e). The generator parameters were set such that tables 31 to 38 are large, 21 to 28 are medium-size tables, and 11 to 18 are the smaller ones. The depth of the hierarchical tree of each instance is six, but tables 16 and 17 (depth is five) and table 18 (depth is four).

For the general tables, whose dimensions are also reported in Table 1, we consid-

Table 1: Characteristics of instances

instance	n	s	m	N. coef.	cont.	bin.	constr.
table 11	20280	973	1560	41314	40560	973	5452
table 12	21476	2062	1684	43784	42952	2062	9932
table 13	36806	3345	5832	76087	73612	3345	19212
table 14	5388	224	1474	11346	10776	224	2370
table 15	26884	2443	3368	54681	53768	2443	13140
table 16	52063	4732	6900	106282	104126	4732	25828
table 17	16852	1531	2390	34551	33704	1531	8514
table 18	8316	755	1339	17204	16632	755	4359
table 21	126362	12324	5501	255102	252724	12324	54797
table 22	43365	4128	3577	88221	86730	4128	20089
table 23	71640	10442	3430	144684	143280	10442	45198
table 24	166248	12927	7966	335808	332496	12927	59674
table 25	55620	4323	2877	112536	111240	4323	20169
table 26	209456	16110	12164	422994	418912	16110	76604
table 27	65241	4744	7240	131780	130482	4744	26216
table 28	88164	6854	4321	178164	176328	6854	31737
table 31	155841	14744	6876	314716	311682	14744	65852
table 32	443169	44096	18230	893718	886338	44096	194614
table 33	116841	14083	4586	235926	233682	14083	60918
table 34	180999	26432	6456	364854	361998	26432	112184
table 35	499298	55527	20747	1007124	998596	55527	242855
table 36	1200439	107743	45638	2417196	2400878	107743	476610
table 37	296004	42652	10904	597057	592008	42652	181512
table 38	572373	81359	18873	1152345	1144746	81359	344309
hier13x13x13d	2197	108	3549	11661	4394	108	3981
hier16	3564	224	5484	19996	7128	224	6380
ninenew	6546	858	7340	32920	13092	858	10772
nine12	10399	1178	11362	52624	20798	1178	16074

ered four complex instances of CSPLIB (a library of tabular data protection instances, available from http://webpages.ull.es/users/casc/#CSPlib:).

4.3 Results

Both 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. 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. 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.

Figures 5 and 6 show the evolution of the feasible solutions obtained for, respectively, the medium-size and the large instances. Results for smallest tables, not shown, revealed a similar pattern, though the differences between the methods were less significant. In those figures, lines "BCD" refer to random-BCD, "B&C" to standard branch-and-cut solutions and "SAT B&C" to branch-and-cut started with SAT solution. This latter option allows to compare BCD and branch-and-cut from the same starting point. 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 CPLEX, which were usually of very poor quality.

Five blocks of cells were considered for the instances 11 to 18 with random-BCD, while ten blocks were used for instances 21 to 28, and 31 to 38. For the instances of Figures 5 and 6, the total time limit was set to three hours. For both BCD variants, the subproblem time limit was one hour. In all cases —but for the 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 larger than three hours, as some BCD subproblems exhausted the subproblem time limit, thus exceeding the total time. This may happen sometimes with random-BCD (as in table 38), but is more usual with tree-BCD because it 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. On the other hand, random-BCD usually consists of a longer sequence of subproblems, which are usually solved faster, though new solutions do not always improve on the previous one.

With regard to the smallest instances 11 to 18, the objective function reaches an almost steady state relatively soon, where BCD variants were faster in achieving an acceptable solution. Only table 14 could be solved by the branch-and-cut method within an optimality tolerance of 5%, and table 18 finished with a gap of just 5.20%. Longer computation times were not of much help: for instance, tables 13 and 16 were run 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

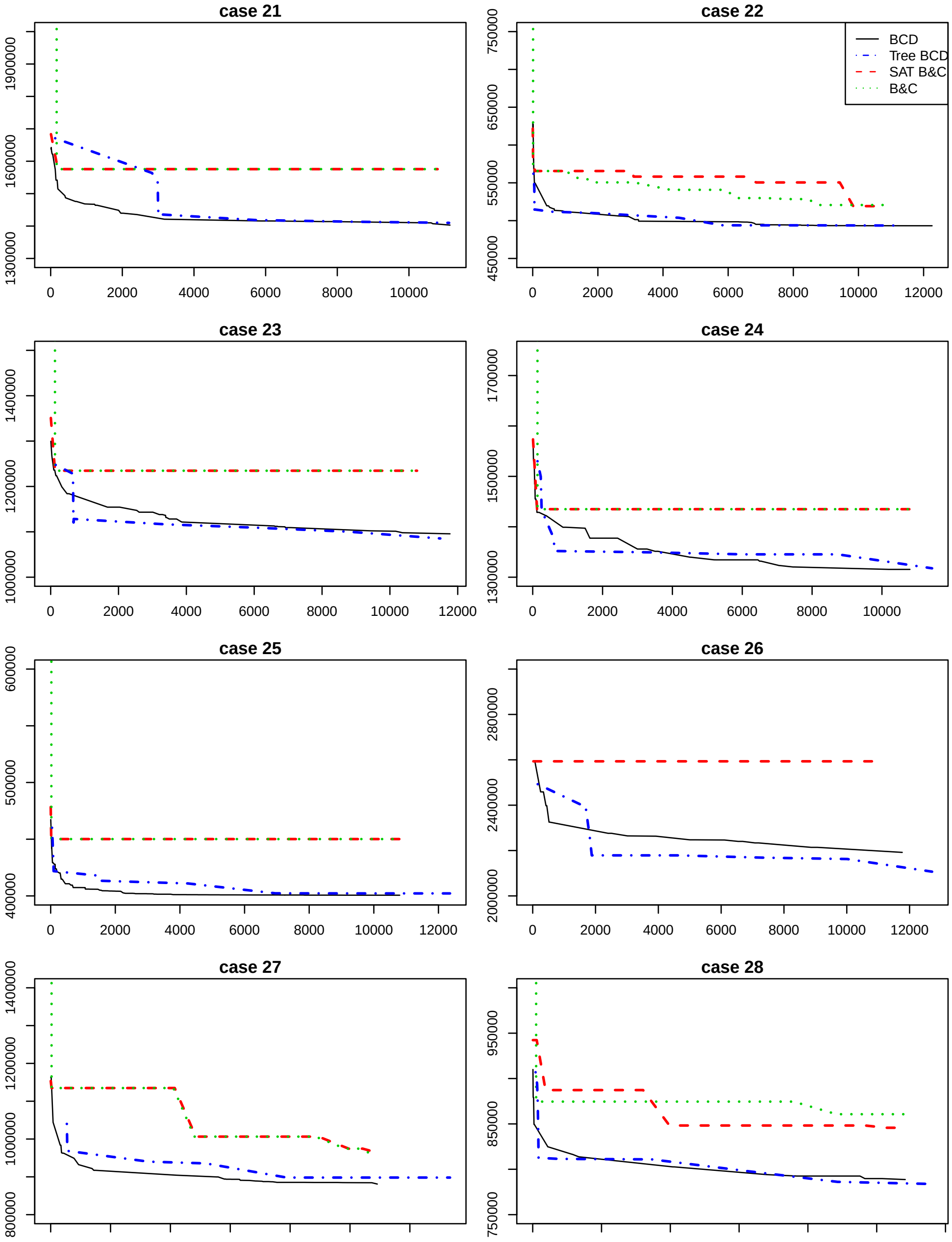


Figure 5: Sequences of feasible solutions for tables 21 to 28. The horizontal axis shows the CPU time and the vertical axis the best objective function value achieved. The time limit was three hours. The branch-and-cut solution for table 26 is not shown because the best value of its objective function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

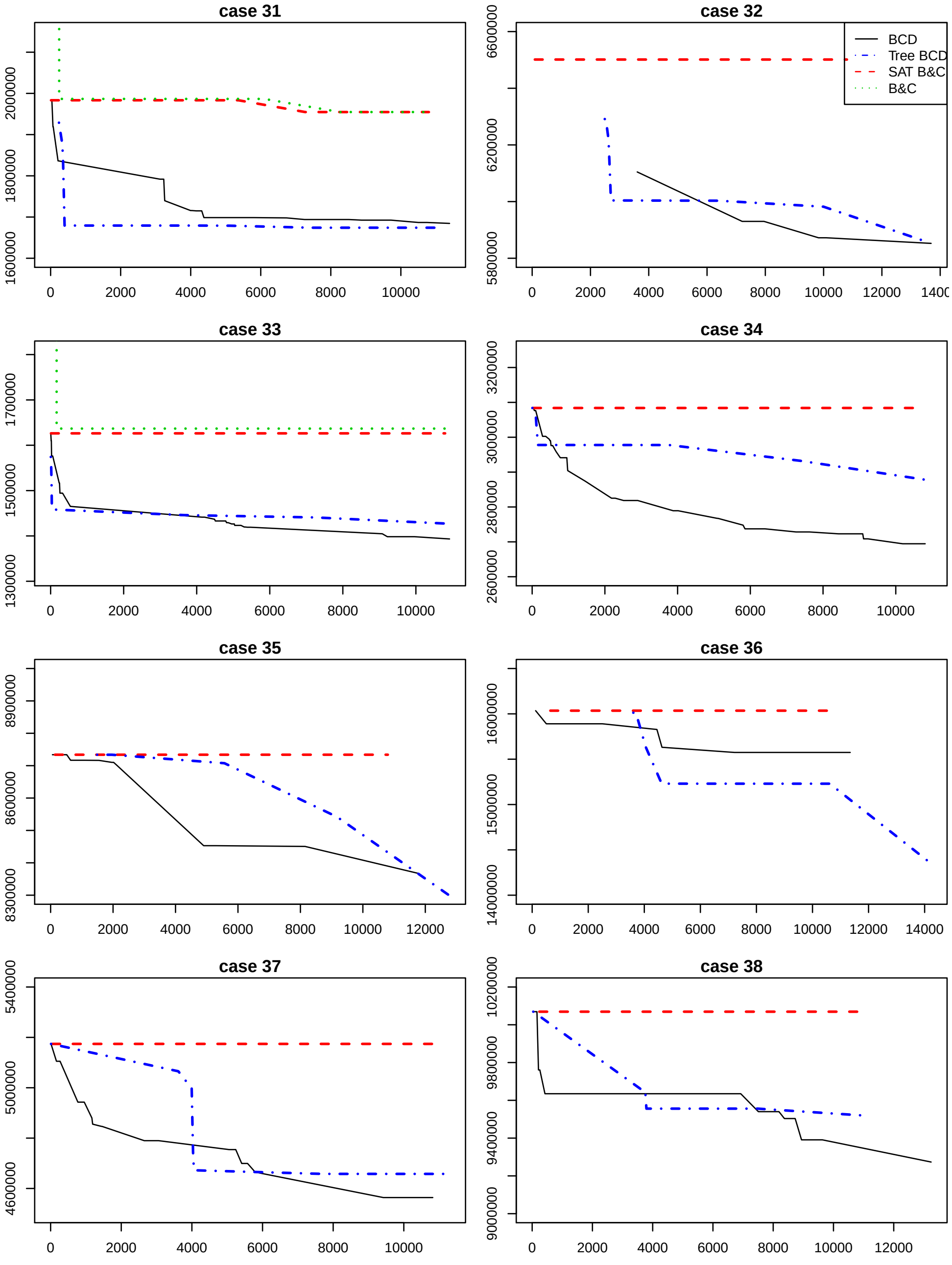


Figure 6: Sequences of feasible solutions for tables 31 to 38. The horizontal axis shows the CPU time and the vertical axis the best objective function value achieved. The branch-and-cut solution for all tables but numbers 31 and 33 are not shown because the best value of their objective functions were about 1000 times the values presented here.

Table 2: 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.

better than those reached by BCD within two hours, the improvement in the objective function is not worth the computational effort.

With larger instances, 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 $5.33 \cdot 10^9$. The poor initial point obtained by the CPLEX strategy may be in the root of the weak performance of the standard branch-and-cut: in Figure 6, only instances 31 and 33 obtain a solution in the same order of magnitude as that of BCD. The SAT proposal is clearly superior in all cases, and it is seen that the evolution of BCD is better than the sequence of branch-and-cut, which usually do not progress within the time window considered.

It is also observed, for instances of any size, that random-BCD and tree-BCD are very similar, the former being slightly better. This was not a priori expected, since tree-BCD was supposed to take advantage of the tree structure of the tables.

Table 2 shows the results obtained for the four complex general tables of Table 1. Columns "B&C" and BCD report the objective function value reached within the CPU time limit shown by column "Time limit" for, respectively, the CPLEX branch-and-cut and random-BCD. From these results, it cannot be concluded that BCD is always 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 much better solution within the time limit.

It is worth noting that above results were obtained with the default options of CPLEX MILP solver, both for the standard branch-and-cut and the BCD approaches. In order to test if the performance of CPLEX could be improved by tuning some its MILP parameters, the group of largest tables was solved again with the following changes: first, parameter MIP Emphasis set to "optimality" (i.e., emphasize optimality over feasibility); second, MIP Emphasis set to "bestbound" (i.e., greater emphasis is set on optimality through moving the best bound); third, parameter FPHEUR (feasibility pump [8]) activated. The random-BCD method was also executed with both choices of the MIP Emphasis parameter. It can be concluded that the default parameters used by CPLEX performed better, since worse objective functions were achieved by the manual tuning. BCD always obtained better solutions than branch-and-cut solution with the manual tuning, though not as good as those with the default parameters. As for the feasibility pump heuristic, it did not outperform the automatic CPLEX strategy for initial points (which may select feasibility pump, if necessary); in any case, points obtained with either the automatic or feasibility pump procedures were always poorer than those computed by the SAT approach.

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 MILP solvers. The BCD approach presented and tested in this work was able to obtain good solutions within two or three hours of the CPU time limit, for large (up to 1,200,000 cells, 100,000 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, even if the initial point found by the SAT procedure is used as warm-start for the branch-and-cut solver. Even tuning some MILP parameters in order to improve the CPLEX performance, the BCD approach consistently returned the best solutions.

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 ones. 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 [2]; some preliminary testing with a prototype showed the approach is efficient for two-dimensional tables [7], 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 grants MTM2009-08747 of the Spanish Ministry of Science and Innovation, and SGR-2009-1122 of the Government of Catalonia.

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] 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.

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

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

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

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

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

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

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

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

[11] 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`).

[12] 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.

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

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

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

[16] 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.

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

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

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

[20] Giessing, S., Höhne, J. (2010), Eliminating small cells from census counts tables: some considerations on transition probabilities, *Lecture Notes in Computer Science*, 6344, 52–65.

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

[22] Glover, F., Cox, L.H., Kelly, J.P., and Patil, R. (2008), Exact, heuristic and metaheuristic methods for confidentiality protection by controlled tabular adjustment, *International Journal of Operations Research* 5, 117–128.

[23] Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Lenz, R., Naylor, J., Schulte Nordholt E., Seri, G., De Wolf, P.-P. (2010), *Handook on Statistical Disclosure Control*, Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control. Available online at http://neon.vb.cbs.nl/casc/SDC_Handbook.pdf.

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

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

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

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

[28] Zayatz, L. (2009), U.S. Census Bureau, communication at Joint UN-ECE/Eurostat Work Session on Statistical Data Confidentiality, Bilbao (Basque Country, Spain).

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