A fix-and-relax heuristic for controlled tabular adjustment

Daniel Baena Jordi Castro José A. González
Dept. of Stat. and Operations Research
Universitat Politècnica de Catalunya
daniel.baena@upc.edu jordi.castro@upc.edu jose.a.gonzalez@upc.edu
Research Report UPC-DEIO DR 2013-02
July, 2013; updated July 2014, October 2014

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

Fix-and-relax approaches for controlled tabular adjustment

Daniel Baena[a], Jordi Castro[*,a], José A. González[a]

[a]*Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Barcelona*

Abstract

Controlled tabular adjustment (CTA) is a relatively new protection technique for tabular data protection. CTA formulates a mixed integer linear programming problem, which is challenging for tables of moderate size. Even finding a feasible initial solution may be a challenging task for large instances. On the other hand, end users of tabular data protection techniques give priority to fast executions and are thus satisfied in practice with suboptimal solutions. This work has two goals. First, the fix-and-relax (FR) strategy is applied to obtain good feasible initial solutions to large CTA instances. FR is based on partitioning the set of binary variables into clusters to selectively explore a smaller branch-and-cut tree. Secondly, the FR solution is used as a warm start for a block coordinate descent (BCD) heuristic (approach named FR+BCD); BCD was confirmed to be a good option for large CTA instances in an earlier paper by the second and third co-authors (Computers & Operations Research 2011). We report extensive computational results on a set of real-world and synthetic CTA instances. FR is shown to be competitive compared to CPLEX branch-and-cut in terms of quickly finding either a feasible solution or a good upper bound. FR+BCD improved the quality of FR solutions for approximately 25% and 50% of the synthetic and real-world instances, respectively. FR or FR+BCD provided similar or better solutions in less CPU time than CPLEX for 73% of the difficult real-world instances.

Key words: Fix-and-Relax, Block Coordinate Descent, Mixed-integer Linear Programming, Controlled Tabular Adjustment, Primal Heuristics, Feasibility Pump, Statistical Disclosure Control

1. Introduction

Microdata and tabular data protection are the two main disciplines of statistical disclosure control. The purpose of this field is to avoid that confidential information can be derived from data released. This is one of the main concerns of National Statistical Agencies (NSAs), which have to disseminate a large amount of information minimizing at the same time the disclosure risk of individual respondents. Tabular data is obtained by crossing two or more categorical variables in a microdata file. For each cell, the table may report either the number of individuals (frequency tables) or information about another variable (magnitude tables). More details can be found in the recent survey [5] and the monographs [26, 27].

[*]Corresponding address: Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Campus Nord, Office C5203, Jordi Girona 1–3, 08034 Barcelona, Catalonia, Spain

Email addresses: daniel.baena@upc.edu (Daniel Baena), jordi.castro@upc.edu (Jordi Castro), jose.a.gonzalez@upc.edu (José A. González)

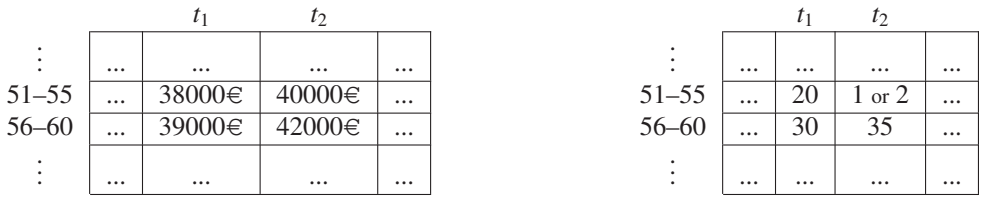

Figure 1: Example of disclosure in tabular data. (a) Average salary per age and town. (b) Number of individuals per age and town. If there is only one individual in town t_2 and age interval 51–55, then any external attacker knows the salary of this single person is 40000€. For two individuals, any of them can deduce the salary of the other, becoming an internal attacker.

Although cell tables report aggregated information for several respondents—so they could be considered anonymized—there is a risk of disclosing individual data. Figure 1 illustrates this situation with a simple case. The left table (a) reports the average salary of individuals by age (row variable) and town (column variable), while table (b) provides the number of individuals. If there were only one individual of age between 51–55 in town t_2, then any external attacker would know the confidential salary of this person. For two individuals, any of them could disclose the other's salary, becoming an internal attacker. Cells that require protection (such as that of the example) are named sensitive, unsafe, or confidential cells. Sensitive cells are a priori detected by some sensitivity rules. The above example showed the simplest *minimum-frequency* rule, which considers sensitive those cells with very few respondents. The most widely used rule, named *p-%* rule, considers a cell unsafe if some respondent may obtain an estimate of another respondent contribution within a *p-%* precision. A detailed description of these rules can be found in [27].

A tabular data protection method can be seen as a map F such that $F(T) = T'$, i.e., table T is transformed to another table T'. Two are the main requirements for F: (1) the output table T' should be "safe", and (2) the quality of T' should be high (or equivalently, the information loss should be small), i.e., T' should be a good replacement for T. The disclosure risk can be analyzed through the inverse map $T = F^{-1}(T')$: if not available or difficult to compute by any *data attacker*, then we may guarantee that F is safe. Controlled Tabular Adjustment (CTA) [3, 11] is a recent technique for the protection of any tabular data. It was empirically observed in [6] that estimates $\hat{T} = \hat{F}^{-1}(T')$, $\hat{F}^{-1}$ being an estimate of F^{-1} for CTA, were not close to T for some real tables. CTA can thus be considered a safe method in general. Moreover, the quality of CTA solutions has shown to be high [10], higher than that provided by alternative methods in some real instances [9].

The goal of CTA—which will be formulated in Section 2—is, given a table with any structure, to find the closest *safe* table to the original one. This is achieved by adding the minimum amount of deviations (or perturbations) to the original cell tables that makes the released table safe. Safety is guaranteed by imposing that sensitive cells in the new protected table are far enough from the original value. This means the cell value is either above *or* below some certain values, thus a disjunctive constraint involving a binary variable is needed for each sensitive cell. The minimum amount of above or below perturbations required for each sensitive cell are named, respectively, *upper protection* and *lower protection* levels. Changes in sensitive cells force other changes in the remaining cells to guarantee that the value of total or marginal cells is preserved.

Although it is a recent approach, CTA is gaining recognition among NSAs; for instance, CTA is considered a relatively new emerging method in the recent monographs [26, 27]. We recently implemented a package for CTA in collaboration with the NSAs of Germany and the Netherlands, within a project funded by Eurostat, the Statistical Office of the European Communities. This package has been largely improved within the FP7-INFRA-2010-262608 project funded by the European Union, with the participation, among others, of the national statistical institutes of Germany, Netherlands, Finland, Sweden and Slovenia. This CTA software is included in the tau-Argus package [25] (`http://neon.vb.cbs.nl/casc/tau.htm`), used for many European national statistical institutes for the protection of tabular data. Among the recent literature on CTA variants we find [8, 24]. In recent specialized workshops on statistical disclosure control, some NSAs stated that perturbative methods, like CTA, are gaining acceptance [31], and perturbative approaches are being used for the protection of national census tables (e.g., [21] for Germany). CTA has also been used within other wider protection schemes, such as the pre-tabular protection method of [20]. In addition, some National Statistical Agencies are questioning current non-perturbative protection methods because "the task of balancing confidentiality and usability [...] is nearly impossible" [30]. Therefore there is a need for new methods, and this justifies the research on CTA and other approaches. Indeed, there is no actually any protection method that fits the needs of all NSAs in the world.

From a computational point of view, the size of the CTA optimization problem is by far smaller than for other well-known protection methods, such as the cell suppression problem [4, 19]. Despite these nice features, CTA formulates a challenging mixed integer linear problem (MILP) for current state-of-the-art solvers (such as CPLEX or XPress). Optimal (or suboptimal, e.g., with a 5% gap) solutions may require many hours of execution for medium instances; very large or massive tables can not be tackled with current technology. Several approaches have been tried to speed up the solution time. A straightforward Benders reformulation of the problem was attempted in [7], but promising results were only obtained for two-dimensional tables (i.e., tables obtained by crossing two categorical variables, whose constraints are represented by a node-arc network incidence matrix [5]). Heuristic and metaheuristic methods were attempted in [22], but they only solved small two-dimensional and three-dimensional tables of up to 625 and 8000 cells, respectively, while we consider in this work much more complex synthetic and real tables from the literature, of up to 200000 and 36000 cells, respectively. For instance, we generated a set of 20 two-dimensional and 20 three-dimensional tables with the same characteristics (sizes and number of sensitive cells) than those in [22]. We remark that: (1) the tables used in [22] were also randomly generated; (2) the matrix constraints only depends on the table structure (two- or three-dimensional table) so they were the same in our experiments and those in [22]; (3) although the instances are not *exactly* the same, what makes difficult (in general) a problem is the structure of the matrix constraints and the number of sensitive cells (which is associated to the number of binary variables of the optimization problem); those characteristics are the same in our experiments and those of [22]. CPLEX 12.5 found a 0% gap solution for all these two-dimensional tables with an average CPU time of 0.02 seconds (the maximum time required by an instance was 0.03 seconds). For the three-dimensional tables, the average CPU time was 0.2 seconds (the maximum time for an instance was 0.49 seconds), again for 0% gap solutions. No CPU time comparison with CPLEX was reported in [22]; it was just stated that CPLEX 8.1 could not solve the instances. Therefore, up to now, there is no conclusive evidence that those metaheuristics are helpful for the CTA problem.

We also tried in the past other general metaheuristics as genetic algorithms without success: combinations or modifications of solutions are not expected to satisfy the large number of linear

constraints with no particular structure of CTA. Indeed, these constraints are usually complex, and any practical approach must rely on the efficient solution of (usually difficult) linearly constrained problems (either LPs or MILPs). The approaches in this paper rely on decomposing the problem into smaller, thus tractable, MILP instances. It is worth to note that even the LPs obtained from large CTA instances by fixing the binary variables are very difficult for today state-of-the-art solvers. Indeed, some of these instances have been included in standard LP repositories [29].

The purpose of this work is twofold. Its first goal is to apply a fix-and-relax (FR) heuristic [13] to the MILP CTA problem. Briefly, FR partitions the set of binary variables into k clusters, and iteratively optimizes for each cluster $i = 1, \ldots, k$, fixing the binary variables of clusters $j < i$ at the optimal value found in previous iterations, and relaxing the integrality of binary variables of clusters $j > i$. The effect of this partitioning of the set of binary variables is that the nodes of the branch-and-cut tree are selectively explored. Equipping this procedure with a backward repartition strategy (details will be given in Section 3.1), if the MILP is feasible then FR will always provide a feasible, hopefully good and efficient, suboptimal solution. The approach cannot guarantee the optimal solution, but in practice end users of statistical data protection techniques prefer quick suboptimal solutions than optimal costly ones, i.e., requiring too many hours, days or weeks of CPU time.

The second objective of the work is to apply a hybrid approach combining FR and the block coordinate descent (BCD) heuristic, which was successfully applied to some classes of CTA problems in [23]. This hybrid method will be named FR+BCD. Indeed, FR is efficient for computing initial, hopefully good, feasible points, while BCD requires a feasible starting point. Therefore, both heuristics are complementary. As it will be shown in Section 4, BCD, warm started with the FR solutions, was able to reduce the gap of the FR solution in approximately half of the real-world CTA instances. In 25 of the 34 real-world instances FR or FR+BCD provided similar or better objective functions in less CPU time than the state-of-the-art MILP solver CPLEX.

FR has been successfully applied in the past mainly to scheduling problems [13, 15, 16]. In those applications, variables and constraints can naturally be partitioned according to some sequential stages, two consecutive ones being only linked by a few of the variables and constraints of each partition. Such a structure can also be found in some classes of tables, named *two dimensional tables with one hierarchical variable*, or, shortly, 1H2D tables. These tables are obtained by crossing a particular categorical variable with a set of, say, h categorical variables that have a hierarchical relation; this results in a set of h two-dimensional tables with some common cells. For instance, Figure 2 (from [5]) illustrates a particular 1H2D table. The left subtable shows number of respondents for "region"×"profession"; the middle subtable is a "zoom in" of region R_2, providing the number of respondents in municipalities of this region; finally the right subtable details the ZIP codes of municipality R_{21}. This type of tables, which are relevant for NSAs, are a priori suitable for FR. Most of the instances tested in the computational results of this work are 1H2D, and, as it will be shown, FR provides good solutions in a fraction of the time required by state-of-the-art branch-and-cut solvers (to obtain equivalent solutions, i.e., with the same objective function value). It will be seen that FR+BCD improved the FR solutions in only 25% of these 1H2D tables. For real-world tables, this percentage increased up to 50%, making FR+BCD a competitive approach.

The paper is organized as follows. Section 2 outlines the MILP CTA problem. Section 3.1 describes the FR heuristic for CTA; it also outlines the BCD approach. Finally, Section 4 presents extensive computational results, showing the effectiveness of FR and FR+BCD for

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

T_1

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

T_2

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

T_3

Figure 2: 1H2D table made of three subtables: "region"×"profession", "municipality"×"profession" and "zip code"×"profession".

synthetic 1H2D and real-world tables.

2. The MILP formulation of the CTA problem

Any CTA problem instance can be represented by the following parameters :

- A set of cells a_i, $i \in \mathcal{N} = \{1, \ldots, n\}$, that satisfies $\mathcal{M} = \{1, \ldots, m\}$ linear relations $Aa = b$, $a \in \mathbb{R}^n$ being the vector of a_i's, and $A \in \mathbb{R}^{m \times n}$. These linear relations impose that the set of inner cells has to be equal to the total or marginal cell, i.e., if $\mathcal{I}_j$ is the set of inner cells of relation $j \in \mathcal{M}$, and t_j is the index of the total cell of relation j, the constraint associated to this relation is $\left(\sum_{i \in \mathcal{I}_j} a_i\right) - a_{t_j} = 0$.
- Nonnegative cell weights w_i, $i \in \mathcal{N}$, used in the definition of the objective function. These weights penalize perturbations from the original cell values in the released table. Cells weights are usually a function of the cell value, e.g., $w_i = 1/a_i$—for this particular weights, the objective function represents relative cell deviations.
- A lower and upper bound for each cell $i \in \mathcal{N}$, respectively l_{a_i} and u_{a_i}, which can be considered publicly known.
- A set $\mathcal{S} = \{i_1, i_2, \ldots, i_s\} \subseteq \mathcal{N}$ of indices of sensitive or confidential cells.
- A lower and upper protection level for each sensitive cell, respectively, lpl_i and upl_i, $i \in \mathcal{S}$. Values of sensitive cells must be out of the interval $(a_i - lpl_i, a_i + upl_i)$ in the released table.

The purpose of CTA is to find the closest safe values x_i to a_i. Considering any distance ℓ, CTA can be formulated as

$$\begin{array}{ll} \min\limits_{x} & \|x - a\|_\ell \\ \text{s. to} & Ax = b \\ & l_{a_i} \le x_i \le u_{a_i} \quad i \in \mathcal{N} \\ & x_i \le a_i - lpl_i \text{ or } x_i \ge a_i + upl_i \quad i \in \mathcal{S}. \end{array} \tag{1}$$

The disjunctive constraints of (1) guarantee the published value is safely out of the interval $(a_i - lpl_i, a_i + upl_i)$. Problem (1) can also be formulated in terms of deviations from the current

cell values. Defining $z_i = x_i - a_i$, $i \in \mathcal{N}$—and similarly $l_{z_i} = l_{a_i} - a_i$ and $u_{z_i} = u_{a_i} - a_i$—, (1) can be recast as

$$\begin{array}{ll} \min\limits_{z} & \|z\|_\ell \\ \text{s. to} & Az = 0 \\ & l_{z_i} \le z_i \le u_{z_i} \quad i \in \mathcal{N} \\ & z_i \le -lpl_i \text{ or } z_i \ge upl_i \quad i \in \mathcal{S}, \end{array} \tag{2}$$

$z \in \mathbb{R}^n$ being the vector of cell deviations. Using the ℓ_1 or Manhattan distance and the cell weights w_i, the objective function is $\sum_{i\in\mathcal{N}} w_i|z_i|$. Since w_i are nonnegative, splitting the vector of deviations z in two nonnegative vectors $z^+ \in \mathbb{R}^n$ and $z^- \in \mathbb{R}^n$, model (2) with the ℓ_1 distance can thus be written as

$$\begin{array}{lll} \min\limits_{z^+,z^-,y} & \sum\limits_{i\in\mathcal{N}} w_i(z_i^+ + z_i^-) & \\ \text{s. to} & A(z^+ - z^-) = 0 & \\ & 0 \le z_i^+ \le \quad u_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\ & 0 \le z_i^- \le -l_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\ & upl_i\, y_i \le z_i^+ \le u_{z_i}\, y_i & i \in \mathcal{S} \\ & lpl_i(1 - y_i) \le z_i^- \le -l_{z_i}(1 - y_i) & i \in \mathcal{S} \\ & y_i \in \{0, 1\}, & i \in \mathcal{S}, \end{array} \tag{3}$$

$y \in \mathbb{R}^s$ being the vector of binary variables associated to protection directions. When $y_i = 1$ the constraints mean $upl_i \le z_i^+ \le u_{z_i}$ and $z_i^- = 0$, thus the protection direction is "upper"; when $y_i = 0$ we get $z_i^+ = 0$ and $lpl_i \le z_i^- \le -l_{z_i}$, thus the protection direction is "lower".

3. Heuristic methods applied to CTA

Model (3) is a difficult MILP even for medium size tables. Finding an optimal (or quasi-optimal) solution may require many hours (even days or weeks) of execution. When the number of sensitive cells is large, the branch-and-cut scheme has shown to be inefficient, and in some cases it is even unable to provide a first feasible solution. For some massive instances—such as, e.g., those in `http://www-eio.upc.es/~jcastro/huge_sdc_instances.html`— the LPs obtained by fixing the value of binary variables—associated to the protection directions—are even not solvable with moderate computational resources. For example, the LPs derived from the six million cells instances of the above web address exhausted the memory of a 16 gigabytes workstation when solved with the CPLEX barrier solver. Unfortunately, the alternative simplex solver is even more prohibitive, but in terms of CPU time: interior-point algorithms have shown to be much more efficient than the simplex for the LPs derived from CTA [3, 5]. In this work we consider a FR heuristic and a hybrid FR+BCD approach for CTA. FR and BCD are, respectively, described and outlined below.

3.1. Fix-and-relax

FR is a decomposition method based on partitioning the set of binary variables into clusters to iteratively solve a sequence of MILPs of smaller dimension than the original problem. In those smaller MILPs only a subset of variables retain their binary constraints while the rest are either fixed or relaxed. Since only a reduced subset of (non-fixed) 0-1 variables is kept integer at each FR iteration, a computational improvement is expected. FR can both be seen as an approach for obtaining (hopefully good) initial feasible solutions and primal bounds. There are

1. Input: Number of clusters $k \geq 1$
2. Partition $\mathcal{S}$ into $\{V_1, \ldots, V_k\}$ clusters
3. Initialize $r = 1$ and solve CTA^1_{FR}
4. **if** CTA^1_{FR} is infeasible, STOP
5. **else** Store values of binary variables of CTA^1_{FR}, set lower bound LB, and $r \leftarrow r + 1$
6. **while** $r \leq k$ **do**
7. Solve CTA^r_{FR}
8. **if** infeasible, redefine the partition structure as in (5)
9. **else** Store optimal values of binary variables of CTA^r_{FR}, and $r \leftarrow r + 1$
10. **end while**
11. Return UB (solution of CTA^k_{FR}) and LB

Figure 3: The fix-and-relax heuristic applied to the CTA problem

other approaches for initial good solutions in MILPs, such as the feasibility pump [17], but as it will seen in Section 4, in practice FR outperformed them.

FR can be briefly stated as follows. The set of binary variables is partitioned into a finite set of clusters $\{V_1, \ldots, V_k\}$. The original MILP is then decomposed into k subproblems and at each iteration one of them is solved. At first iteration (counter r set to 1) the subproblem considers as binary only the variables of V_1, while the integrality of binary variables in the remaining clusters is relaxed. Continuous variables in the original MILP maintain this same status at each subproblem. Hopefully, this first subproblem will be easily solved since the cardinality of V_1 is much smaller than the number of binary variables in the original MILP. Once solved, the counter r is incremented and the next subproblem is considered. At subproblem of iteration r, $k > r > 1$, the binary variables of clusters V_i, $i < r$, are fixed to the values of optimal solutions from the previous iterations; variables of cluster V_r are considered binary, while the integrality of variables in clusters V_j, $j > r$ is relaxed. The process is repeated until $r = k$. If no subproblem is infeasible, a (hopefully good) feasible solution will be available after the solution of subproblem k. In the particular case of CTA, the set $\mathcal{S}$ of sensitive cells is partitioned into the subsets $\{V_1, \ldots, V_k\}$, and the subproblem r associated to (3)—which will be referred as (CTA^r_{FR})—is

$$
\begin{array}{lll}
\min\limits_{z^+, z^-, y} & \sum\limits_{i=1}^{n} w_i (z_i^+ + z_i^-) & \\
\text{s. to} & A(z^+ - z^-) = 0 & \\
& 0 \leq z_i^+ \leq \quad u_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\
& 0 \leq z_i^- \leq -l_{z_i} & i \in \mathcal{N} \setminus \mathcal{S} \\
& upl_i \, y_i \leq z_i^+ \leq u_{z_i} \, y_i & i \in \mathcal{S} \\
& lpl_i (1 - y_i) \leq z_i^- \leq -l_{z_i} (1 - y_i) & i \in \mathcal{S} \\
& y_i = \tilde{y}_i & i \in \bigcup_{h=1,\ldots,r-1} V_h \\
& y_i \in \{0, 1\} & i \in V_r \\
& y_i \in [0, 1] & i \in \bigcup_{h=r+1,\ldots,k} V_h,
\end{array}
\tag{4}
$$

where $\tilde{y}_i$, $i \in \cup_{h=1,\ldots,r-1} V_h$, are the values of binary variables found at subproblems CTA^1_{FR}, …, CTA^{r-1}_{FR}. Although FR is a heuristic for MILP problems, it is easily switched to an optimal approach by setting $k = 1$.

It is worth noting that the first subproblem CTA^1_{FR} has two main features compared to the subsequent ones:

- The lower bound on the objective function provided by CTA^1_{FR} is a global lower bound of (3). On the other hand, the lower bound of subproblems $r > 1$ are just local lower bounds. The lower bound reported by the FR algorithm will then be that of CTA^1_{FR}. Note that the optimal solution of CTA^1_{FR} can be considered a lower bound of (3) *only* if computed with a 0% gap. However, such a gap is impractical, because the solution of CTA^1_{FR} would take a long execution time—something to avoid, since the goal of FR is to quickly provide a decent solution. In the implementation developed, the lower bound was obtained by the CPLEX routine CPXgetbestobjval. When a problem has been solved to optimality, this routine provides the optimal solution value. Otherwise, it provides the minimum objective function value of all remaining unexplored nodes in the branch-and-cut tree.

- If CTA^1_{FR} is infeasible, then (3) is infeasible as well. However, if some subproblem $r > 1$ is infeasible it can not be concluded that (3) is infeasible; it just means that we can not fix $y_i = \tilde{y}_i$, for $i \in V_{r-1}$, at subproblem r. To overcome this drawback, when subproblem $r > 1$ is reported as infeasible, we *backtrack* to problem $r-1$, modifying the partition by joining the clusters V_{r-1} and V_r as follows:

$$\begin{cases} V_{r-1} \leftarrow V_{r-1} \bigcup V_r \\ V_i \leftarrow V_{i+1}, i = r, \ldots, k-1 \\ k \leftarrow k-1 \\ r \leftarrow r-1. \end{cases} \tag{5}$$

  Note that the above repartition strategy will always provide a feasible solution if (3) is feasible. Indeed, in the worst case, if subproblem k is infeasible and (5) is applied $k-1$ times, we will end up with a unique cluster, i.e., we will be solving (3). However, in practice, as it was observed in the computational results of Section 4, this repartition strategy was never needed in the instances tested.

An outline of the FR algorithm for CTA is shown in Figure 3.

3.2. Outline of block coordinate descent

The BCD approach applied to CTA has been described in [23]. Briefly, it consists of a sequence of CTA subproblems, each of them optimizing the objective function over the cell deviations z^+, z^- and a subset of the decision variables y, while the remaining variables y are kept fixed to some direction. Provided that we start from a feasible assignment of y, the method can move from a solution to another, hopefully better. Although there could be uncountable strategies to determine the subset of variables to be optimized, the set $\mathcal{S}$ is usually partitioned into k clusters (or blocks) and the algorithm iterates through them. However, BCD could perform indefinitely, starting again with the same or with another partition. Stopping criteria normally employed are: only one cycle of k clusters; a time limit, or a specified number of subproblems without improvement in the objective function. Since the method does not account for dual information there are no means to compute a gap for the solution. Despite this, the results of [23] showed that BCD reaches sub-optimal but still good solutions in significantly less time than branch-and-cut schemes. The algorithm is summarized in Figure 4.

Experience with BCD has shown that, in general, the performance of the method improves as the number of blocks decreases, and two blocks seems to be the best choice. Notice that one block would lead the method to a plain branch-and-cut, which might be computationally prohibitive. It has been observed that many tables are (sub-optimally) protected through manipulation of half

1. Input: Number of clusters $k \geq 1$
2. Set feasible initial values to y; initialize outer iteration counter: $t \leftarrow 0$
3. **while** stopping criterion not satisfied and $t \leq t_{\max}$ **do**
4. Set inner iteration counter: $i \leftarrow 0$; divide y into k blocks: $y = \{y^{1,i}, \ldots, y^{k,i}\}$
5. **while** $i < k$ **do**
6. Solve (3) with respect to block $y^{i,i}$ fixing $y^{j,i}$ for $j \neq i$: obtain $(y^{i,i})^*$
7. $y^{i,i+1} \leftarrow (y^{i,i})^*$; $y^{j,i+1} \leftarrow y^{j,i}$ for $j \neq i$
8. $i \leftarrow i + 1$
9. **end while**
10. $t \leftarrow t + 1$
11. **end while**
12. Return the best solution found

Figure 4: The block coordinate descent heuristic for the CTA problem

of their sensitive cells in a fraction of the time needed if the whole set of sensitive cells was considered (this fraction of time being significantly less than $1/2$).

Many tests indicate that rebuilding the partition of blocks at each iteration is clearly preferable to keep some pre-determined division. Actually, the best performances are obtained with a random division of the binary variables into blocks; this is the main strategy considered.

A disadvantage of BCD is the need to find a feasible initial assignment of directions to start the process (step 2 of algorithm of Figure 4), which may be in itself a difficult problem for large CTA instances. The heuristic approach considered in [23], which relies on the Boolean Satisfiability problem, only focuses on the constraints, and then it may provide poor quality solutions. Since FR solutions take into account the objective function, we can use them as a good warm start to BCD. This approach, named FR+BCD, will be computationally tested and seen as a very efficient option in Subsection 4.3.

4. Computational results

The FR and FR+BCD heuristics for CTA have been coded in C++, using the state-of-the-art CPLEX 12.5 branch-and-cut solver for the solution of subproblems (4). FR and FR+BCD were compared with the direct solution of (3) through plain CPLEX branch-and-cut, which will be referred as BC.

All the runs were carried out on a 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. Default values were used for the CPLEX parameters, unless explicitly stated. For the computational tests we considered a set of real-world general and synthetic 1H2D tables. Real-world general tables are standard instances used in the literature [5]. It is worth noting that some real-world instances were not included in this set since they are too difficult for both heuristic and exact MILP approaches—no feasible solution was obtained within the time limit. Synthetic instances were obtained with a generator of 1H2D tables. This generator is governed by several parameters, as, for instance, the number of rows in a subtable; the number of columns per subtable; the depth of the hierarchical tree; the minimum and maximum number of rows with hierarchies for each subtable; and the probability for a cell to be marked as sensitive. The 1H2D table generator is available from `http://www-eio.upc.es/~jcastro/generators_csp.html`. We

instance	$\overline{n}$	$\overline{s}$	$\overline{m}$	$\overline{nz}$
Symmetric instances				
sym-40-50-5	29039	1421	1334	58793
sym-40-50-15	31753	46612	1388	64219
sym-40-50-30	29141	8556	1336	58997
sym-40-60-5	36990	1816	1521	74835
sym-40-60-15	34026	5011	1473	68906
sym-40-60-30	38040	11207	1539	76933
sym-50-50-5	40637	1989	1562	81988
sym-50-50-15	39596	5815	1541	79907
sym-50-50-30	38097	11190	1512	76908
sym-50-60-5	45555	2237	1662	91964
sym-50-60-15	44457	6550	1644	89768
sym-50-60-30	45835	13507	1666	92525
Asymmetric instances				
asym-40-50-5	125661	6157	5677	254483
asym-40-50-15	126844	18646	5700	256850
asym-40-50-30	127000	37338	5703	257162
asym-40-60-5	151166	7431	6321	306114
asym-40-60-15	149641	22069	6296	303064
asym-40-60-30	150711	44454	6314	305203
asym-50-50-5	162561	7966	6400	328284
asym-50-50-15	159766	23487	6346	322694
asym-50-50-30	160171	47094	6354	323503
asym-50-60-5	191503	9415	6982	386789
asym-50-60-15	189718	27982	6953	383218
asym-50-60-30	188742	55676	6937	381266

Table 1: Characteristics of symmetric/asymmetric synthetic 1H2D instances.

fixed all parameters, but three: the number of rows per subtable ($\mathtt{r} \in \{40, 50\}$), the number of columns per subtable ($\mathtt{c} \in \{50, 60\}$) and the percentage of sensitive cells ($\mathtt{s} \in \{5, 15, 30\}$).

We considered either symmetric and asymmetric instances, i.e., instances where $u_{a_i} = l_{a_i}$ for all $i \in \mathcal{N}$ and $u_{a_i} \neq l_{a_i}$ for some $i \in \mathcal{N}$, respectively. Asymmetric instances were obtained by considering $u_{a_i} = \mathtt{a} \cdot l_{a_i}$ for all $i \in \mathcal{N}$, where $\mathtt{a} \in \{2, 5, 10\}$ is the asymmetry parameter. For each combination of parameters we generated a sample of five instances varying the random generator seed. This amounted to 12 and 36 samples of five instances each one, for symmetric and asymmetric instances respectively. Although the asymmetry parameter slightly affects to the difficulty of the problem, both symmetric and asymmetric instances will be grouped by `r`, `c` and `s` to simplify the exposition. The reported computational results are thus averaged on five and 15 replications for symmetric and asymmetric tables, respectively.

Table 1 reports the characteristics of each set of symmetric/asymmetric 1H2D instances: the average number of cells ("$\overline{n}$"), the average number of sensitive cells ("$\overline{s}$"), the average number of table relations ("$\overline{m}$") and the average number of coefficients in linear constraints ("$\overline{nz}$"). Hierarchical synthetic tables are identified by the particular combination of parameters, i.e., sym-`r`-`c`-`s` for symmetric instances and asym-`r`-`c`-`s` for asymmetric ones. Table 2 reports the same information for real-world tables, though in this case the dimensions are not averaged. The dimensions of the MILP problems (3) are $2n$ continuous variables, s binary variables, and $m + 4s$ linear constraints.

4.1. Tuning the number of clusters in fix-and-relax

The performance of FR depends on the number of clusters k considered. We performed an empirical study of the effect of k on two particular metrics: the CPU time and the quality of the solutions provided by FR. This empirical analysis was done considering values $k \in \mathcal{K} =$

instance	n	s	m	nz
australia_ABS	24420	918	274	13224
bts4	36570	2260	36310	136912
cbs	11163	2467	244	22326
dale	16514	4923	405	33028
destatis	5940	621	1464	18180
hier13d4	18969	2188	47675	143953
hier13	2020	112	3313	11929
hier13x13x13a	2197	108	3549	11661
hier13x13x13b	2197	108	3549	11661
hier13x13x13c	2197	108	3549	11661
hier13x13x13d	2197	108	3549	11661
hier13x13x13e	2197	112	3549	11661
hier13x13x7d	1183	75	1443	5369
hier13x7x7d	637	50	525	2401
hier16	3564	224	5484	19996
hier16x16x16a	4096	224	5376	21504
hier16x16x16b	4096	224	5376	21504
hier16x16x16c	4096	224	5376	21504
hier16x16x16d	4096	224	5376	21504
hier16x16x16e	4096	224	5376	21504
nine5d	10733	1661	17295	58135
osorio	10201	7	202	20402
sbs2008_C	4212	1135	2580	13806
sbs2008_E	1430	382	991	4680
table1	1584	146	510	4752
table3	4992	517	2464	19968
table4	4992	517	2464	19968
table5	4992	517	2464	19968
table6	1584	146	510	4752
table7	624	17	230	1872
table8	1271	3	72	2542
targus	162	13	63	360
toy3dsarah	2890	376	1649	9690
two5in6	5681	720	9629	34310

Table 2: Characteristics of real instances.

$\{3, 5, 7, 10, 20, 30, 40, 50\}$, and using a subset of asymmetric 1H2D instances. For each combination of parameters `r-c-s` we considered a sample of three instances.

Each instance was solved $|\mathcal{K}|$ times, randomly partitioning the set $\mathcal{S}$ of sensitive cells into $k \in \mathcal{K}$ subsets. The stopping criterion for all the runs, i.e., subproblems (4), was a 5% optimality gap, which is computed by CPLEX as $(\text{UB} - \text{LB})/(|\text{UB}| + 10^{-10})$, where UB is the best integer solution (upper bound) and LB is the best achievable value from the current branch-and-cut tree (lower bound).

Figure 5 reports the CPU time (in seconds, averaged for the three instances of each sample) used by FR for the different $k \in \mathcal{K}$ number of clusters. Clearly, the CPU time increases with k, and the heuristic becomes prohibitive if the number of clusters is large.

The second metric, the quality of the solutions, was evaluated using the performance profile proposed in [14]. Quality was measured as the value of the objective function (thus, the lower, the better). Let Q_{tk} be the quality of the solution of instance t solved by FR with k clusters. Note that Q_{tk} for CTA is always strictly positive. The performance ratio is thus defined as

$$v(t, k) = \frac{Q_{t,k}}{\min\{Q_{t,k} : k \in \mathcal{K}\}},$$

i.e., the ratio between the quality of the solution obtained when instance t is solved by FR with k clusters over the strategy with the best (minimum) performance for this instance. The (cumulative) distribution function $P_k(q) : [1, \infty) \to [0, 1]$ is defined as

$$P_k(q) = \frac{|\{t \in \mathcal{T} : v(t, k) \leq q\}|}{|\mathcal{T}|}, q \geq 1.$$

where $\mathcal{T}$ is the set of instances. Figure 6 shows the performance profiles for the different $k \in \mathcal{K}$. $P_k(q) = 1$ means FR with k clusters is able to solve all the instances within a factor q of the best possible ratio. In our case $k = 3$ is the first strategy to converge to 1 for $q \approx 1.45$ (i.e., FR with 3 blocks solves all the instances within a factor ≈ 1.45 of the best ratio). It can also be observed that $k = 3$ provides the highest quality for 80% of the instances ($P_3(1) \approx 0.8$).

4.2. Comparison between fix-and-relax and plain branch-and-cut

From the discussion of previous Subsection, $k = 3$ was set for FR. An optimality gap of 5% was considered for all the optimization problems, either (3) or FR subproblems (4). The time limit was set to two hours for both 1H2D and real-world instances. Note that FR subproblems are also solved by CPLEX branch-and-cut; therefore the comparison is between whether using or not the FR scheme. We will refer to these two variants as FR and BC.

Tables 3 and 4 report an exhaustive comparison between FR and BC for synthetic 1H2D and real-world instances, respectively. These tables report the FR CPU time (columns "T_{FR}"); the primal gap of the solution reported by FR (columns "$GAP_{FR}\%$"); the primal gap of the solution reported by BC after T_{FR} seconds of CPU time (columns "$GAP_{BC}\%$"), i.e., using the same time than FR; the difference between both primal gaps (columns "$\Delta(BC, FR)$"); the primal gap and CPU time needed by BC to compute a better solution than the feasible solution found by FR (columns "$GAP^{up}_{BC}\%$" and "T^{up}_{BC}"); and finally the difference between the time needed by BC to improve the FR solution and the time needed by FR to compute that solution (columns"$\Delta(T_{FR}, T^{up}_{BC})$"). Positive values at column $\Delta(BC, FR)$ means that FR achieved a better solution than BC in the same CPU time.

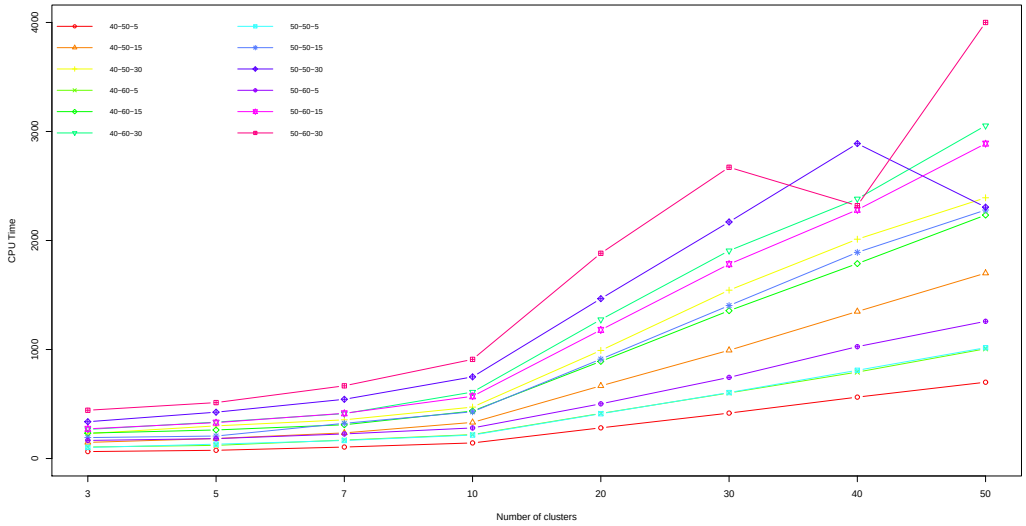

Figure 5: CPU time for different number of clusters

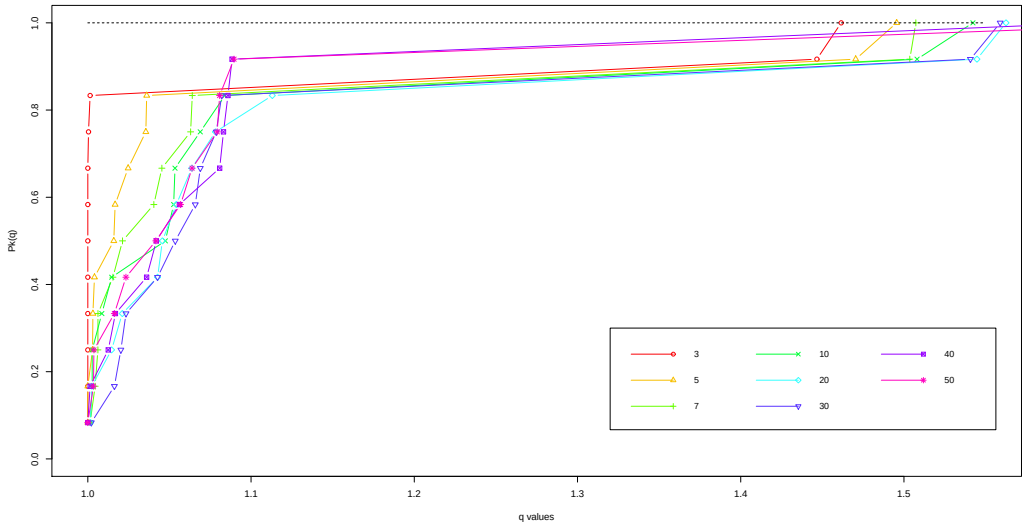

Figure 6: Performance profile of the quality of the solution for different number of clusters

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC,FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T^{up}_{BC},T_{FR})$
asym-40-50-5	72.76	2.52	$\dagger^{(40.70,5)}$	$\dagger^{(38.18,5)}$	$\ddagger^{(1.20,13)}$	$\ddagger^{(87.76,13)}$	$\ddagger^{(15.00,13)}$
asym-40-50-15	158.75	3.55	86.67	83.11	$\ddagger^{(1.94,13)}$	$\ddagger^{(465.84,13)}$	$\ddagger^{(307.09,13)}$
asym-40-50-30	210.51	5.60	99.98	94.38	1.84	1083.47	872.96
asym-40-60-5	95.40	2.40	$\dagger^{(0.88,5)}$	$\dagger^{(-1.51,5)}$	$\ddagger^{(1.08,14)}$	$\ddagger^{(124.01,14)}$	$\ddagger^{(28.61,14)}$
asym-40-60-15	193.37	3.17	99.96	96.79	$\ddagger^{(1.69,12)}$	$\ddagger^{(945.40,12)}$	$\ddagger^{(752.03,12)}$
asym-40-60-30	314.96	5.26	99.97	94.71	2.43	1107.84	792.88
asym-50-50-5	110.54	1.82	$\dagger^{(0.08,8)}$	$\dagger^{(-1.74,8)}$	$\ddagger^{(0.62,14)}$	$\ddagger^{(135.65,14)}$	$\ddagger^{(25.11,14)}$
asym-50-50-15	186.95	3.11	86.66	83.54	$\ddagger^{(1.02,11)}$	$\ddagger^{(804.02,11)}$	$\ddagger^{(617.07,11)}$
asym-50-50-30	333.50	5.94	99.94	93.99	2.21	1704.06	1370.55
asym-50-60-5	153.14	1.23	$\dagger^{(0.45,9)}$	$\dagger^{(-0.78,9)}$	$\ddagger^{(0.85,13)}$	$\ddagger^{(163.61,13)}$	$\ddagger^{(10.46,13)}$
asym-50-60-15	282.79	3.05	93.33	90.28	$\ddagger^{(1.46,13)}$	$\ddagger^{(1569.41,13)}$	$\ddagger^{(1286.62,13)}$
asym-50-60-30	406.42	5.52	99.92	94.40	2.39	1396.89	990.47
sym-40-50-5	8.99	2.43	12.69	10.26	$\ddagger^{(1.61,2)}$	$\ddagger^{(15.09,2)}$	$\ddagger^{(6.10,2)}$
sym-40-50-15	115.34	4.31	45.79	41.48	$\ddagger^{(2.56,4)}$	$\ddagger^{(443.91,4)}$	$\ddagger^{(328.57,4)}$
sym-40-50-30	371.35	4.61	63.90	59.29	$\ddagger^{(4.36,3)}$	$\ddagger^{(2681.80,3)}$	$\ddagger^{(2310.45,3)}$
sym-40-60-5	10.52	2.79	14.44	11.65	1,33	37.76	27.24
sym-40-60-15	102.29	2.20	82.23	80.03	$\ddagger^{(-,0)}$	$\ddagger^{(-,0)}$	$\ddagger^{(-,0)}$
sym-40-60-30	800.45	4.39	12.88	8.49	$\dagger^{(4.59,3)}$	$\dagger^{(2870.02,3)}$	$\dagger^{(2069.57,3)}$
sym-50-50-5	25.47	1.94	$\dagger^{(12.20,3)}$	$\dagger^{(10.25,3)}$	$\ddagger^{(0.72,5)}$	$\ddagger^{(50.75,5)}$	$\ddagger^{(25.27,5)}$
sym-50-50-15	166.33	3.66	12.74	9.08	$\ddagger^{(1.95,2)}$	$\ddagger^{(1434.04,2)}$	$\ddagger^{(1267.71,2)}$
sym-50-50-30	511.19	3.45	66.81	63.36	$\ddagger^{(3.71,2)}$	$\ddagger^{(5049.30,2)}$	$\ddagger^{(4538.11,2)}$
sym-50-60-5	56.80	1.61	$\dagger^{(54.33,4)}$	$\dagger^{(52.72,4)}$	$\ddagger^{(0.97,4)}$	$\ddagger^{(104.60,4)}$	$\ddagger^{(47.80,4)}$
sym-50-60-15	279.63	2.47	13.60	11.14	$\ddagger^{(0.70,1)}$	$\ddagger^{(1055.10,1)}$	$\ddagger^{(775.47,1)}$
sym-50-60-30	833.45	3.95	47.62	43.66	$\ddagger^{(4.38,2)}$	$\ddagger^{(3863.53,2)}$	$\ddagger^{(3030.08,2)}$

$\dagger^{(x,y)}$ BC could not find a solution in y of the overall number of replications within T_{FR} seconds; x is the average value for the remaining successful runs.

$\ddagger^{(z,w)}$ BC could not improve the FR solution in w of the overall number of replications within the time limit; z is the average value for the remaining successful runs.

Table 3: Comparison between fix-an-relax and plain branch-and-cut for synthetic asymmetric and symmetric 1H2D instances

From Table 3 it can be concluded that FR is more efficient than BC for fast good feasible solutions of 1H2D tables. In several runs (marked with ‡) BC could not find a better solution than FR within the time limit. It is worth noting that for all the 1H2D instances FR provided solutions with gaps below 6%. For the real-world general instances of Table 4 the situation is slightly different. These instances are not guaranteed to have a hierarchical structure, and this may explain why FR is not as competitive as for 1H2D tables. FR provided a better gap than BC within the same CPU time in 17 of the 34 instances, and both FR and BC provided the same gap in six adittional cases. In six of these cases BC could not improve the FR solution within the two hours time limit. In the remaining instances BC outperformed FR.

4.3. Comparison between fix-and-relax with block coordinate descent and plain branch-and-cut

As mentioned in section 3.2, since FR can provide good feasible solutions faster in average than BC, BCD was warm started with the FR solution. This hybrid approach was named FR+BCD.

The BCD algorithm performed in all cases a loop with two clusters, each one with a half of the sensitive cells, partitioned at random. At exit, the CPU computation time and the objective function value were saved. This CPU time was added to the FR CPU time and compared to the CPU time used by BC. We also took into account whether the sensitive cells had been correctly protected in the final solutions, since accuracy errors might be present in some instances, making actually infeasible the protected table. This accuracy errors are due to the big-M constraints $z_i^+ \leq u_{z_i}\, y_i$ and $z_i^- \leq -l_{z_i}(1-y_i)$ of (3) and (4), since u_{z_i} and $-l_{z_i}$ can take very large values.

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC, FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T^{up}_{BC}, T_{FR})$
australia_ABS	6,05	73,87	3,84	-70,03	7,40	2,45	-3,6
bts4	6332,15	66,57	74,16	7,59	‡	‡	‡
cbs	2,87	100,00	100,00	0,00	0,00	2,88	0,01
dale	595,85	48,44	48,44	0,00	48,44	7199,96	6604,11
destatis	204,32	19,53	99,97	80,44	1,80	706,48	502,16
hier13d4	6410,73	82,86	99,98	17,12	‡	‡	‡
hier13	747,29	6,88	4,90	-1,98	4,90	159,24	-588,05
hier13x13x13a	542,72	5,22	5,22	0,00	4,94	690,15	147,43
hier13x13x13b	584,92	5,89	5,24	-0,65	5,24	369,13	-215,79
hier13x13x13c	542,86	5,63	4,99	-0,65	4,99	243,43	-299,43
hier13x13x13d	178,12	4,69	5,27	0,58	2,40	340,47	162,35
hier13x13x13e	336,72	5,39	4,40	-0,99	4,40	269,82	-66,9
hier13x13x7d	34,72	5,58	7,19	1,61	4,96	69,32	34,6
hier13x7x7d	2,71	4,82	12,35	7,53	‡	‡	‡
hier16	4854,46	59,48	63,07	3,59	‡	‡	‡
hier16x16x16a	2803,76	44,96	48,80	3,84	44,71	3706,65	902,89
hier16x16x16b	3401,2	33,49	99,95	66,46	31,89	6275,87	2874,67
hier16x16x16c	3488,27	40,86	50,40	9,54	39,62	4926,13	1437,86
hier16x16x16d	3776,81	57,66	63,32	5,66	57,07	5369,03	1592,22
hier16x16x16e	3862,21	46,55	46,87	0,32	‡	‡	‡
nine5d	6133,25	67,69	99,99	32,30	‡	‡	‡
osorio	1,86	0,00	0,00	0,00	0,00	1,03	-0,83
sbs2008_C	543,88	50,11	3,36	-46,75	21,77	32,24	-511,64
sbs2008_E	4,56	4,73	4,73	0,00	4,73	2,94	-1,62
table1	0,62	8,38	13,43	5,06	4,92	1,25	0,63
table3	1909,18	25,39	15,07	-10,32	17,11	408,3	-1500,88
table4	1196,86	25,39	17,30	-8,09	18,60	511,64	-685,22
table5	720,49	22,80	20,20	-2,60	20,25	216,19	-504,3
table6	1,01	7,77	39,32	31,54	3,36	2,17	1,16
table7	0,1	1,01	0,41	-0,61	0,41	0,02	-0,08
table8	0,18	2,44	0,00	-2,44	1,35	0,04	-0,14
targus	0,08	3,84	3,84	0,00	3,84	0,01	-0,07
toy3dsarah	25,47	0,34	7,29	6,94	0,34	37,46	11,99
two5in6	3010,89	66,04	99,99	33,95	63,91	7200,45	4189,56

‡ Time limit reached without improving the feasible FR solution.

Table 4: Comparison between fix-an-relax and plain branch-and-cut for real instances

			Objective Function	
			FR+BCD < BC	FR+BCD > BC
	FR+BCD < BC			
		mean (sd) ΔF	−1.24 (1.08)	1.86 (1.46)
		max $\lvert\Delta F\rvert$	−4.23	7.26
		mean (sd) ΔT	−1564 (1872)	−883 (1281)
		max $\lvert\Delta T\rvert$	−6641	−6351
		N [N_{asym} ; N_{sym}]	58 [27; 31]	119 [93; 26]
Time	FR+BCD > BC			
		mean (sd) ΔF	−1.55 (1.43)	0.83 (1.07)
		max $\lvert\Delta F\rvert$	−3.89	4.17
		mean (sd) ΔT	146 (207)	49 (38)
		max $\lvert\Delta T\rvert$	508	182
		N [N_{asym} ; N_{sym}]	5 [4; 1]	58 [56; 2]

ΔF stands for $100(F_{FR+BCD} - F_{BC})/F_{BC}$. ΔT stands for $(T_{FR+BCD} - T_{BC})$, in seconds.
"sd" stands for standard deviation.

Table 5: Summary of results for 1H2D instances, in the comparison between FR+BCD versus BC.

With regard to 1H2D instances, it has been observed that the extra time needed by the BCD stage is related to the number of sensitive cells, although with considerable variability especially if the table is large. The 16 tables with more than 50,000 sensitive cells consumed between 114 and 492 seconds, with a median time of 255 seconds. In 104 instances with less than 10,000 sensitive cells the median time was 29.7 seconds. Compared to the time employed by the FR stage, it took about 40% of that time (median proportion): in 18 instances out of 240 BCD lasted longer than FR, generally in tables with high density of sensitive cells.

Sixty-one tables improved the objective function after the BCD stage, and the others remained in the same value (not necessarily in the same solution). The median change in the objective function with respect to the value attained by FR was 3%, with a maximum of 10%. Improving the solution requires also more time: 54.6% of FR time, instead of 37% for the tables not improved. We observed a higher rate of success among the tables with high density of sensitive cells: an odd of 33 versus 47 for tables with 30% of sensitive cells, compared to 28 versus 132 for tables with 15% or lower proportion. The table size or the asymmetry degree in the protection levels were not related to improvement in the objective function.

Table 5 summarizes the results with 1H2D instances for two factors: solution times (in rows) and objective function values (in columns) between BC and FR+BCD. The two categories for each factor are either FR+BCD outperfomed BC ("FR+BCD < BC", i.e., less CPU time or a lower objective function value) or the opposite ("FR+BCD > BC"). Each of the four cells shows the number of instances ("N"), and some statistics (mean, standard deviation, maximum) about the change in both factors: Δ_F is the percentage change in the objective function, Δ_T is the absolute change in the time. Comparing left versus right columns, we can see small differences between the percentage changes in objective function values (they range from −4.23% to 7.26%). However, comparing above versus below rows, we can see large differences with respect to solution times: 1564 and 883 seconds in favor of FR+BCD (177 cases) against 146 and 49 seconds (63 cases) in favor of BC.

For the real-world tables, FR+BCD got better solutions than FR in 18 instances after a BCD cycle, whereas it did not improve the FR objective function in 16 cases. Table 6 reports the results obtained. Columns "$F_.$" and "$T_.$" provide, respectively, the objective function and CPU solution times for each method, BC, FR and FR+BCD or BCD. Column "$\Delta(F_{FR}, F_{FR+BCD})$" provides the relative change (as a percentage) in the objective function between the FR and FR+BCD solutions. The rows are ordered by $\Delta(F_{FR}, F_{FR+BCD})$; the first instance shows a negative change

instance	F_{BC}	F_{FR}	F_{FR+BCD}	T_{BC}	T_{FR}.	T_{BCD}	$\Delta(F_{FR}, F_{FR+BCD})$
table6‡	28331962414	29686800000	29899600000	2.17	1	0.38	-0.72
dale	256	256	256	7199.96	595.9	596.62	0
hier13	434834824.5	444063000	444063000	1312.7	747.3	8.18	0
hier13d4	5.11488e+12	6143970000	6143970000	7201.23	6410.7	2769.53	0
hier13x13x13a	434834824.5	436127000	436127000	895.44	542.7	6.16	0
hier13x13x13b	44385.67	44865.8	44865.8	1444.94	584.9	7.14	0
hier13x13x13c	368036.2	370564	370564	1561.69	542.9	8.47	0
hier13x13x13d	414115.44	424074	424074	340.48	178.1	6.89	0
hier13x13x13e	4644973.87	4693570	4693570	269.83	336.7	6.83	0
hier13x7x7d	594401	593370	593370	29.19	2.7	0.24	0
hier16	591756145.8	556221000	556221000	7200.41	4854.5	130.81	0
hier16x16x16b	74891.53	76700.4	76700.4	7200.44	3401.2	121.52	0
osorio	13	13	13	1.8	1.9	1.35	0
sbs2008_E	109959.57	109960	109960	2.95	4.6	0.13	0
table7	9970266227	10031200000	10031200000	0.04	0.1	0.1	0
toy3dsarah	5.0747e+14	5.07506e+14	5.07506e+14	37.46	25.5	0.22	0
hier13x13x7d	1684140	1695250	1686430	241.81	34.7	2.59	0.52
table8	439	450	445	0.09	0.2	0.18	1.11
hier16x16x16a	529703489.9	532145000	525466000	7200.31	2803.8	372.61	1.26
targus	1103759.75	1103760	1088480	0.02	0.1	0.08	1.38
nine5d	6.20788e+12	1215790000	1191810000	7200.51	6133.3	2389.56	1.97
table5	10154665.5	11094800	10837600	7200.21	720.5	258.66	2.32
hier16x16x16c	604844.28	620633	601548	7200.39	3488.3	672.13	3.08
table4	10290147784	11843300000	11433900000	7200.27	1196.9	142.92	3.46
hier16x16x16e	9543201.06	9485820	9077750	7200.41	3862.2	692.34	4.3
table1	2.93185e+13	3.04227e+13	2.89576e+13	1.45	0.6	0.54	4.82
two5in6	707133564.8	751514000	713214000	7200.45	3010.9	169.86	5.1
hier16x16x16d	752648610.3	765577000	725869000	7200.31	3776.8	789.54	5.19
table3	1.20849e+12	1.39866e+12	1.29248e+12	7200.24	1909.2	171.17	7.59
destatis	234541294	286199000	241329000	2528.91	204.3	29.83	15.68
bts4	4114851966	3180710000	2592590000	7200.6	6332.2	6002.29	18.49
sbs2008_C	320835.46	621448	459655	66.34	543.9	0.43	26.03
australia_ABS	651	2396	746	2.91	6.05	4.46	68.86
cbs†	0	268	0	2.88	2.9	1.79	100

‡ This negative improvement is due to unprotected cells in FR solution.
† cbs instance has a global optimum of zero because all the sensitive cells have null weights in the objective function.

Table 6: Comparison between plain branch-and-cut and FR+BCD with real-world instances.

because the solution reached by FR was actually not feasible, due to slight deviations in some sensitive cells beyond their protection levels, but undetectable with the (already tight) infeasibility tolerance in use by the solver (cf. big-M issue discussed above). In general, FR already provided a good solution for instances which could not be improved by BCD; this FR solution was close to the one obtained by BC, but it was computed faster. On the other hand, it is remarkable that most of the instances where the BCD cycle could improve the solution were difficult for the BC scheme, which used to exhaust the time limit.

Table 7 summarizes the results for the real-world instances with respect to CPU time and objective function values. The structure of this table is similar to that of Table 5, but with an additional central column. This central column corresponds to instances without relevant differences in the objective function value (i.e., $(F_{FR+BCD} - F_{BC})/F_{BC}$ less than 5%). Each cell of Table 7 reports the number and names of its instances. The first row includes the instances that were solved faster with FR+BCD than with BC, and the instances that could not be solved in the 2-hour time limit by BC, but they could by FR+BCD. Moreover, some instances were not suitably protected: bts4, dale, table1, table3, table4, table5, table6 and table7 present some sensitive cells unprotected in the BC solution; dale, table5 and table6 had the same problem with FR, and BCD was in trouble as well with bts4 and table6: in general, FR+BCD dealt better than BC with these difficult instances.

		Objective Function		
		FR+BCD < BC	FR+BCD ≈ BC	FR+BCD > BC
Time	FR+BCD < BC	bts4 hier16 nine5d table3 table4 table5 [N=6]	dale destatis hier13 hier13d4 hier13x13x13a hier13x13x13b hier13x13x13c hier13x13x13d hier13x13x7d hier13x7x7d hier16x16x16a hier16x16x16b hier16x16x16c hier16x16x16d hier16x16x16e table1 table6 toy3dsarah two5in6 [N=19]	[N=0]
	FR+BCD > BC	[N=0]	cbs hier13x13x13e osorio sbs2008_E table7 table8 targus [N=7]	australia_ABS sbs2008_C [N=2]

Table 7: Summary of results for real instances, in the comparison between FR+BCD versus BC.

Nine tables were solved faster with the pure BC scheme, but it is worth noting that only two (sbs2008_C and hier13x13x13e) can be considered as challenging, since they needed more than one minute to be solved, whereas four (osorio, table7, table8 and targus) have few sensitive cells and could be solved very quickly by both FR+BCD and BC.

To sum up, Table 7 shows that the combination FR+BCD is competitive with BC in the solution's quality, and, in addition, it protects the table in significantly less time.

4.4. Comparison between fix-and-relax and other heuristics

Current state-of-the-art MILP solvers can be turned into heuristic approaches by tuning some of their pre-build heuristics. For a fair comparison, FR is tested in this section against feasibility pump (FP), relaxation induced neighborhod search (RINS), and FR+BC (warm starting CPLEX from the FR solution) with and without polishing.

4.4.1. Fix-and-relax and feasibility pump heuristics

FP [17] is considered an efficient heuristic for the fast computation of hopefully good initial feasible solutions to MILPs. We used the *objective feasibility pump* (oFP) [1], which is more efficient than FP in terms of quality of the solution and the *analytic center feasibility pump* (AC-FP) [2], which was introduced as a good alternative in some MILP instances (either in time or quality of the solution). Table 8 shows a comparison between FR and these FP variants for real instances. It reports the primal gap of the FR and FP solutions (columns "$GAP_{FR}\%$" and "$GAP_{FP}\%$", respectively), the CPU time required by FR and FP to compute the feasible solution (columns "T_{FR}" and "T_{FP}", respectively), and the difference between both methods in CPU times and gaps (columns "$\Delta(T_{FP}, T_{FR})$" and "$\Delta(FP, FR)$", respectively). We ran both oFP and AC-FP. Table 8 only shows the result of the best FP variant, i.e., the one that provides the lowest gap, and in case of equal gaps, the fastest one. The best FP variant is clearly marked in the table.

instance	$GAP_{FR}\%$	T_{FR}	$GAP_{FP}\%$	T_{FP}	$\Delta(T_{FP}, T_{FR})$	$\Delta(FP, FR)$
australia_ABS	73,87	6,05	$95{,}90^{AC-FP}$	26	19,95	22,03
bts4	66,57	6332,15	$74{,}09^{oFP}$	552	-5780,15	7,52
cbs	100,00	2,87	$100{,}00^{oFP}$	20	17,13	0,00
dale	48,44	595,85	$98{,}52^{oFP}$	27	-568,85	50,08
destatis	19,53	204,32	$21{,}93^{AC-FP}$	222	17,68	2,40
hier13d4	82,86	6410,73	†	†	†	†
hier13	6,88	747,29	$58{,}96^{oFP}$	126	-621,29	52,08
hier13x13x13a	5,22	542,72	$63{,}36^{AC-FP}$	122	-420,72	58,14
hier13x13x13b	5,89	584,92	$53{,}36^{AC-FP}$	235	-349,92	47,47
hier13x13x13c	5,63	542,86	$54{,}01^{oFP}$	217	-325,86	48,38
hier13x13x13d	4,69	178,12	$99{,}86^{oFP}$	132	-46,12	95,17
hier13x13x13e	5,39	336,72	$99{,}87^{oFP}$	128	-208,72	94,48
hier13x13x7d	5,58	34,72	$60{,}49^{AC-FP}$	13	-21,72	54,91
hier13x7x7d	4,82	2,71	$73{,}56^{oFP}$	2	-0,71	68,73
hier16	59,48	4854,46	$68{,}36^{AC-FP}$	2852	-2002,46	8,89
hier16x16x16a	44,96	2803,76	$99{,}99^{oFP}$	4537	1733,24	55,03
hier16x16x16b	33,49	3401,2	$99{,}91^{oFP}$	3742	340,8	66,42
hier16x16x16c	40,86	3488,27	$99{,}93^{oFP}$	3937	448,73	59,07
hier16x16x16d	57,66	3776,81	$66{,}88^{AC-FP}$	2706	-1070,81	9,22
hier16x16x16e	46,55	3862,21	$81{,}19^{oFP}$	4430	567,79	34,64
nine5d	67,69	6133,25	†	†	†	†
osorio	0,00	1,86	$27{,}65^{oFP}$	0	-1,86	27,65
sbs2008_C	50,11	543,88	$82{,}64^{oFP}$	12	-531,88	32,53
sbs2008_E	4,73	4,56	$74{,}15^{oFP}$	2	-2,56	69,42
table1	8,38	0,62	$2{,}17^{oFP}$	0	-0,62	-6,20
table3	25,39	1909,18	$100{,}00^{oFP}$	323	-1586,18	74,61
table4	25,39	1196,86	$96{,}81^{AC-FP}$	379	-817,86	71,42
table6	7,77	1,01	$9{,}05^{oFP}$	0	-1,01	1,28
table7	1,01	0,1	$69{,}21^{oFP}$	0	-0,1	68,20
table8	2,44	0,18	$6{,}51^{oFP}$	0	-0,18	4,07
targus	3,84	0,08	$0{,}92^{oFP}$	0	-0,08	-2,92
toy3dsarah	0,34	25,47	$65{,}07^{oFP}$	5	-20,47	64,73
two5in6	66,04	3010,89	$61{,}91^{AC-FP}$	5234	2223,11	-4,13

† Time limit reached without finding a feasible feasibility pump solution.
^{oFP}: best solution provided by oFP.
$^{AC-FP}$: best solution provided by AC-FP.

Table 8: Comparison between fix-and-relax and feasibility pump for real instances.

It is clearly seen that FR outperformed FP for CTA in terms of quality of the solution. In most cases, FR provided a better gap than FP by a big difference. Only in three instances FP was better. FP reached the time limit without a feasible solution in two instances. However, FP is in general faster than FR in order to find a feasible solution. It can be concluded that, for the CTA problem, FR instead of FP should be used for finding good feasible solutions within a reasonable short time.

4.4.2. Fix-and-relax and RINS and local branching heuristics

RINS [12] is a heuristic that explores a neighborhood of the current incumbent solution and the continuous relaxation at a node h of the BC tree to try to find a new and improved incumbent. CPLEX BC incorporates RINS, allowing the user to control how often to apply the heuristic through a frequency parameter f. A value $f > 0$ means that RINS is applied at nodes $h = 0, f, 2f, ...$ while for $f = 0$ CPLEX automatically decides when to apply the heuristic. The results of Table 4 were obtained with $f = 0$; as it was shown in that table, FR outperformed BC in a considerable percentage of real-world instances. Table 9 adds a comparison between FR and BC with $f = 50$. The meaning of columns is the same as in Table 4. The value of f, either 0 or 50, is reported in the new column $RINS_f$. We only considered the subset of real-world

instance	T_{FR}	$GAP_{FR}\%$	$RINS_f$	$GAP_{BC}\%$	$\Delta(BC, FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T_{FR}, T^{up}_{BC})$
bts4	6332,15	66,57	0	74,16	7,59	‡	‡	‡
			50	100	33,43	‡	‡	‡
dale	595,85	48,44	0	48,44	0	48,44	7199,96	6604,11
			50	48,44	0	‡	‡	‡
destatis	204,32	19,53	0	99,97	80,44	1,8	706,48	502,16
			50	99,97	80,44	1,78	3090,16	2885,84
hier13	747,29	6,88	0	4,9	-1,98	4,9	159,24	-588,05
			50	99,98	93,1	4,9	1239,3	492,01
hier13x13x13a	542,72	5,22	0	5,22	0	4,94	690,15	147,43
			50	99,99	94,77	4,94	1426,84	884,12
hier13x13x13b	584,92	5,89	0	5,24	-0,65	5,24	369,13	-215,79
			50	99,98	94,09	4,87	2742,67	2157,75
hier13x13x13c	542,86	5,63	0	4,99	-0,65	4,99	243,43	-299,43
			50	99,98	94,34	4,99	3405,16	2862,3
hier13x13x7d	34,72	5,58	0	7,19	1,61	4,96	69,32	34,6
			50	38,93	33,34	4,84	158,96	124,24
hier13x7x7d	2,71	4,82	0	12,35	7,53	‡	‡	‡
			50	24,1	19,28	4,53	20,09	17,38
hier16	4854,46	59,48	0	63,07	3,59	‡	‡	‡
			50	100	40,52	‡	‡	‡
hier16x16x16a	2803,76	44,96	0	48,8	3,84	44,71	3706,65	902,89
			50	100	55,03	43,87	7200,57	4396,81
hier16x16x16d	3776,81	57,66	0	63,32	5,66	57,07	5369,03	1592,22
			50	100	42,33	56,08	7200,59	3423,78
hier16x16x16e	3862,21	46,55	0	46,87	0,32	‡	‡	‡
			50	99,96	53,41	46,18	7200,52	3338,31
table3	1909,18	25,39	0	15,07	-10,32	17,11	408,3	-1500,88
			50	100	74,61	16,78	2303,34	394,16
table4	1196,86	25,39	0	17,3	-8,09	18,6	511,64	-685,22
			50	100	74,61	14,64	2163,97	967,11
table5	720,49	22,8	0	20,2	-2,6	20,25	216,19	-504,3
			50	100	77,2	16,96	1441,96	721,47

‡ Time limit reached without improving the feasible fix-and-relax solution.

Table 9: Comparison between FR and BC with frequency RINS f equal to 0 and 50 for some real-world instances

instances whose BC tree had more than 50 nodes. From Table 9 it can be concluded that FR still outperforms BC with the RINS heuristic using $f = 50$.

We additionally tried RINS frequencies $f \in \{100, 150, 200\}$, obtaining *exactly* the same results (they are thus omitted in Table 9). As stated above, CPLEX always applies the RINS heuristic at node 0 for any $f > 0$. We noted that, since RINS is an expensive heuristic, it exhausted most of the allowed time (that of the FR heuristic) at node 0, making irrelevant the particular value of f. Therefore, at least for this particular application, RINS $f = 0$ seems to be the best choice. Indeed, we noted that when $f = 0$ CPLEX does not apply RINS to node 0 in many instances.

The local branching (LBr) heuristic also explores the neighborhood of an incumbent solution, but by adding constraints based on the number of binary variables flipping their values with respect the incumbent [18]. Running CPLEX with the LBr heuristic, and setting as time limit the CPU time of FR, we only observed differences with RINS $f = 0$ for five instances of Table 9: hier13x13x7d (solutions of 6.2% and 7.2% gaps for LBr and RINS, respectively), hier13x7x7d (24.1% gap for LBr, $12,4\%$ gap for RINS), hier16 (61.2% for LBr, 63.0% for RINS), hier16x16x16a (44.4% for LBr, 49.0% for RINS), and hier16x16x16d (62.4% for LBr, $63,0\%$ for RINS). LBr only clearly outperformed RINS $f = 0$ in hier16x16x16a; for that instance, LBr was also more efficient than FR.

instance	T_{FR}	$GAP_{FR}\%$	T_{FR+BC}	$GAP_{FR+BC}\%$	T_{BC}	$GAP_{BC}\%$	$\Delta(FR+BC, BC)$	$\Delta(T_{FR+BC}, T_{BC})$
asym-40-50-5	72.76	1.89	264.45	0.47	306.46	0.35	0.12	−42.01
asym-40-50-15	158.75	3.13	†(1223.55,1)	0.71	†(1357.15,3)	0.67	0.04	−133.60
asym-40-50-30	**210.51**	**5.41**	†**(1606.53,2)**	**0.82**	†**(2062.43,3)**	**1.04**	**−0.23**	**−455.90**
asym-40-60-5	95.40	1.75	259.40	0.40	231.36	0.40	0.00	28.04
asym-40-60-15	**193.37**	**3.11**	†**(1352.60,2)**	**0.93**	†**(1430.58,4)**	**1.02**	**−0.09**	**−77.98**
asym-40-60-30	314.96	5.07	†(2181.63,6)	1.40	†(2040.04,6)	1.21	0.19	141.59
asym-50-50-5	110.54	1.53	†(560.24,1)	0.43	476.49	0.35	0.08	83.75
asym-50-50-15	**186.95**	**2.86**	†**(1403.85,3)**	**0.90**	†**(1533.78,4)**	**0.93**	**−0.03**	**−129.93**
asym-50-50-30	333.50	5.81	†(2417.91,5)	1.52	†(2284.80,5)	1.74	−0.22	133.11
asym-50-60-5	153.14	1.06	268.43	0.64	286.51	0.48	0.16	−18.08
asym-50-60-15	**278.72**	**2.73**	†**(1263.52,3)**	**1.24**	†**(1408.72,3)**	‡**(8.67,2)**	**−7.43**	**−145.21**
asym-50-60-30	416.11	5.54	†(2768.88,8)	1.60	†(2665.53,7)	‡(1.74,1)	−0.14	103.35
sym-40-50-5	8.99	2.00	41.51	0.40	29.10	0.68	−0.28	12.40
sym-40-50-15	115.34	4.21	1114.50	0.75	720.22	0.80	−0.06	394.29
sym-40-50-30	371.35	4.62	†(3097.98,4)	1.93	†(3131.25,3)	1.77	0.16	−33.27
sym-40-60-5	10.52	2.68	32.82	0.63	60.05	0.52	0.11	−27.23
sym-40-60-15	102.29	2.16	1731.73	0.86	1381.75	0.82	0.04	349.98
sym-40-60-30	**800.45**	**4.40**	†**(3600,5)**	**3.22**	†**(3600,5)**	**6.55**	**−3.33**	**0**
sym-50-50-5	25.47	1.88	103.45	0.50	85.12	0.57	−0.07	18.33
sym-50-50-15	166.33	3.66	†(2370.98,1)	0.97	†(1728.63,1)	0.75	0.22	642.35
sym-50-50-30	**511.19**	**3.45**	†**(3600,5)**	**2.54**	†**(3600,5)**	**4.46**	**−1.92**	**0**
sym-50-60-5	56.80	1.59	80.64	0.54	134.79	0.36	0.17	−54.15
sym-50-60-15	279.63	2.46	2347.58	0.91	1894.95	0.74	0.17	452.63
sym-50-60-30	**833.45**	**3.93**	†**(3600,5)**	**3.54**	†**(3600,5)**	**6.31**	**−2.77**	**0**

†$^{(x,y)}$ a solution within 1% optimality gap could not be found in y of the overall number of replications within the time limit of 3600 seconds; x is the average CPU time for the remaining successful runs.

‡$^{(z,w)}$ no feasible solution was found in w of the overall number of replications within the time limit of 3600 seconds; z is the average gap for the remaining runs.

Table 10: Using the fix-and-relax solution to warm start CPLEX branch-and-cut

4.5. Using fix-and-relax to warm start branch-and-cut

Table 10 shows the results obtained with FR+BC (i.e., warm starting BC with the FR solution) on 1H2D tables. The table reports the CPU computation time and gap (as a percentage) of the FR solution (columns "T_{FR}" and "$GAP_{FR}\%$"). The same information is provided for the FR+BC solution using a 1% optimality gap (columns "T_{FR+BC}" and $GAP_{FR+BC}\%$); and for CPLEX BC without starting point with the same 1% optimality gap (columns T_{BC} and $GAP_{BC}\%$). Columns $\Delta(FR+BC, BC)$ and $\Delta(T_{FR+BC}, T_{BC})$ give the difference in gap and CPU time between the FR+BC and BC solutions. A time limit of one hour was considered for these runs. Some FR+BC or BC executions were unable to find a solution of 1% optimality gap within this time limit; these are clearly marked in Table 10. BC could not find a feasible solution within the time limit for three instances, which are also clearly marked in the table. In those situations the average gap reported in Table 10 may be greater than 1%.

FR+BC provided a lower gap than BC in 12 of 24 cases. In addition, in six of these 12 cases the CPU time of FR+BC was inferior. These six successful FR+BC executions are marked in boldface in Table 10. These results are not entirely satisfactory, since it could be expected that providing a good incumbent from the beginning would significantly reduce the computational burden for all the instances, by pruning portions of the search space. In fact, we found reported similar experiences. In `http://www2.isye.gatech.edu/~rcarvajal3/2012/2012-12-24_effect-of-informatio` the author presents an experiment with instances from MIPLIB 2010 [28] where providing the optimal solution as a warm start can actually be harmful for the performance of the solver.

We also applied the CPLEX polishing heuristic to the FR starting point. This heuristic, which can be very time consuming, tries to exploit an initial feasible solution provided to BC by

solving an alternative branch-and-cut. We ran FR+BC with and withoug polishing. Activating the polishing the gap was improved in 92% of the executions; however the average gap reduction was 0.4%. On the other hand, in 83% of the executions the polishing significantly increased the CPU time: an average increment of 59%. In the remaining 17% of executions the CPU time was reduced, in average, a 18%. From these figures, it can be concluded that the polishing is in general very time consuming for CTA, and it is not worth the gap reduction provided.

5. Conclusions

FR, either alone or in combination with other heuristics such as BCD, has shown to be an efficient approach for the difficult MILP CTA problem. Initially developed for scheduling problems that can be partitioned into stages, FR has also been successfully applied to a class of hierarchical tables named 1H2D. For these tables, it was competitive against BC, and FP or RINS heuristics. For general real-world tables, FR and FR+BCD outperformed BC in 73% of the instances tested. Promising results were also obtained in a reduced set of instances by warm starting BC with the FR solution.

Quick tools to provide fast solutions to CTA are a necessity because of the increasing ability of NSAs to create more complex and huge tables from collected data. FR is thus an step in this direction. Combining FR with other heuristics, or embedding FR in exact approaches, like Benders reformulation, is part of the further work to be done in this field.

Acknowledgments

This work has been supported by grant MTM2012-31440 of the Spanish Government. We thank two anonymous reviewers for their comments which greatly improved the presentation of the paper.

References

[1] T. Achterberg, T. Berthold, Improving the feasibility pump, Discrete Optimization 4, 77–86, (2007).
[2] D. Baena, J. Castro, Using the analytic center in the feasibility pump, Operations Research Letters, 39, 310–317 (2011).
[3] J. Castro, Minimum-distance controlled perturbation methods for large-scale tabular data protection, European Journal of Operational Research, 171, 39–52 (2006).
[4] J. Castro, A shortest-paths heuristic for statistical data protection in positive tables, INFORMS Journal on Computing, 19(4), 520–533 (2007).
[5] J. Castro, Recent advances in optimization techniques for statistical tabular data protection, European Journal of Operational Research, 21, 257–269 (2012).
[6] J. Castro, On assessing the disclosure risk of controlled adjustment methods for statistical tabular data, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 20, 921–941 (2012) .
[7] J. Castro, D. Baena, Using a mathematical programming modeling language for optimal CTA, Lecture Notes in Computer Science, 5262, 1–12 (2008).
[8] J. Castro, A. Frangioni, C. Gentile, Perspective reformulations of the CTA problem with L_2 distances, Operations Research, 62(4), 891–909 (2014).
[9] J. Castro, S. Giessing, Testing variants of minimum distance controlled tabular adjustment. In: Monographs of Official Statistics, Eurostat-Office for Official Publications of the European Communities, Luxembourg, 333–343 (2006).
[10] J. Castro, J.A. González, Assessing the information loss of controlled adjustment methods in two-way tables, Lecture Notes in Computer Science, 8744, in press (2014).
[11] R.A. Dandekar, L.H. Cox, Synthetic tabular data: an alternative to complementary cell suppression, manuscript, Energy Information Administration, U.S. Department of Energy (2002).

[12] E. Danna, E. Rothberg, C. Le Pape, Exploring relaxation induced neighborhoods to improve MIP solutions, Mathematical Programming, 102, 71-90 (2005).
[13] C. Dillenberger, L.F. Escudero, A. Wollensak, W. Zhang. On practical resource allocation for production planning and scheduling with period overlapping setups, European Journal of Operational Research, 75, 275–286 (1994).
[14] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming, A, 91, 201–13 (2002).
[15] L.F. Escudero, J. Salmerón, On a fix-and-relax framework for a class of project scheduling problems, Annals of operations research, 140, 163–188 (2005).
[16] D. Ferreira, R. Morabito, S. Rangel, Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants, Computers & Operations Research, 37, 684–691 (2010).
[17] M. Fischetti, F. Glover, A. Lodi, The Feasibility Pump, Mathematical Programming, 104, 91–104 (2005).
[18] M. Fischetti, A. Lodi, Local branching, Mathematical Programming, 98, 23–47 (2003).
[19] M. Fischetti, J.J. Salazar, Solving the cell suppression problem on tabular data with linear constraints, Management Science, 47, 1008–1026 (2001).
[20] S. Giessing, Pre-tabular perturbation with controlled tabular adjustment: some considerations, Lecture Notes in Computer Science, 8744, 48–61 (2014).
[21] S. Giessing, J. Höhne, Eliminating small cells from census counts tables: some considerations on transition probabilities, Lecture Notes in Computer Science, 6344, 52–65 (2010).
[22] F. Glover, L.H. Cox, R. Patil, J.P. Kelly, Integrated exact, hybrid and metaheuristic learning methods for confidentiality protection, Annals of Operations Research, 183, 47–73 (2011).
[23] J.A. González, J. Castro, A heuristic block coordinate descent approach for controlled tabular adjustment, Computers & Operations Research, 38, 1826–1835 (2011).
[24] M.S. Hernández, J.J. Salazar, Enhanced controlled tabular adjustment, Computers & Operations Research, 43, 61–67 (2014).
[25] A. Hundepool, The Argus software in CENEX, Lecture Notes in Computer Science, 4302, 334–346 (2006).
[26] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, R. Lenz, J. Naylor, E. Schulte Nordholt, G. Seri, P.-P. De Wolf, Handook on Statistical Disclosure Control v1.2, Network of Excellence in the European Statistical System in the field of Statistical Disclosure Control, 2010. Available online at `http://neon.vb.cbs.nl/casc/handbook.htm`.
[27] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. Schulte Nordholt, K. Spicer, P.-P. De Wolf, Statistical Disclosure Control. Chichester, Wiley, 2012.
[28] T. Koch, et al., MIPLIB 2010. Mixed integer programming library version 5, Mathematical Programming Computation, 3, 103–163 (2011).
[29] H. Mittelmann, Decision tree for optimization software, http://plato.asu.edu/guide.html (2014).
[30] K. Soininvaara, T. Oinonen, A. Nissinen, Balancing confidentiality and usability: protecting sensitive data in the case of inward Foreign AffiliaTes Statistics (FATS), Lecture Notes in Computer Science, 8744, 338–349 (2014).
[31] L. Zayatz, U.S. Census Bureau, communication at Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality, Bilbao (Basque Country, Spain) (2009).