A fix-and-relax heuristic for controlled tabular adjustment

Daniel Baena Jordi Castro
Dept. of Stat. and Operations Research
Universitat Politècnica de Catalunya
daniel.baena@upc.edu jordi.castro@upc.edu
Research Report UPC-DEIO DR 2013-02
July, 2013

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

A fix-and-relax heuristic for controlled tabular adjustment

Daniel Baena[a], Jordi Castro[*,a]

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

Abstract

Controlled tabular adjustment (CTA) is an emerging protection technique for tabular data protection. CTA formulates a mixed integer linear programming problem, which is tough for tables of moderate size. Finding a feasible initial solution may even 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. In this work the fix-and-relax strategy is applied to large CTA instances. Fix-and-relax is based on partitioning the set of binary variables into clusters to selectively explore a smaller branch-and-cut tree. We report extensive computational results on a set of real and random CTA instances. Fix-and-relax is shown to be competitive compared to plain CPLEX branch-and-cut in terms of quickly finding either a feasible solution or a good upper bound in difficult instances.

Key words: Fix-and-Relax, 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 no confidential information can be derived from data released. This is one of the main concerns of National Statistical Agencies (NSA's), 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 monograph [16].

Although cell tables report aggregated information for several respondents, 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. Even if the number of respondents was larger, this cell could be considered unsafe if

*Corresponding address: Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Campus Nord, Office C5218, Jordi Girona 1–3, 08034 Barcelona, Catalonia, Spain

Email addresses: daniel.baena@upc.edu (Daniel Baena), jordi.castro@upc.edu (Jordi Castro)

July 25, 2013

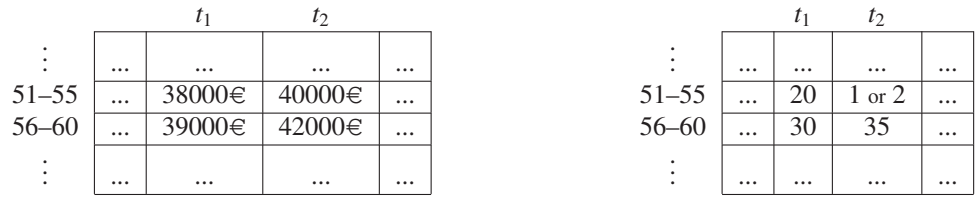

Figure 1: Example of disclosure in tabular data. (a) Average salary per age and town. (b) Number of individuals per age and town. If there is only one individual in town t_2 and age interval 51–55, then any external attacker knows the salary of this single person is 40000€. For two individuals, any of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

one individual could obtain a good estimator of another's salary (for instance, by subtracting its own contribution from the cell value). Cells that require protection are named unsafe, sensitive or confidential cells. Controlled Tabular Adjustment (CTA) [3] is a recent technique for the protection of any tabular data. The goal of CTA—which will be formulated in Section 2—is to add the minimum amount of deviations to the original cell tables to obtain the safe table which is closest to the original one. Although it is a recent approach, CTA is gaining recognition among NSAs; for instance, CTA is considered as an emerging method in the recent monograph [16]. 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. In recent specialized workshops on statistical disclosure control, some NSAs stated that perturbative methods, like CTA, are gaining acceptance [18], and perturbative approaches are being used for the protection of national census tables (e.g., [14] for Germany).

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 (CSP) [4, 13]. Moreover, the quality of CTA solutions has shown to be higher than that provided by CSP in some real instances, exhibiting a low disclosure risk [6]. 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 approaches based on block coordinate descent (BCD) have also been applied to some classes of tables [15].

The purpose of this work is to apply a fix-and-relax heuristic [8] to the MILP CTA problem. Briefly, fix-and-relax 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), if the MILP is feasible then fix-and-relax will always provide a feasible, hopefully good and efficient, suboptimal solution. The approach cannot guarantee the optimal

	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".

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.

Fix-and-relax has been successfully applied in the past mainly to scheduling problems [8, 10, 11]. 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}. 1H2D is a relevant type of tables for NSAs, which is a priori suitable for fix-and-relax. Most of the instances tested in the computational results of this work are 1H2D, and, as it will be shown, fix-and-relax 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).

The paper is organized as follows. Section 2 outlines the MILP CTA problem. Section 3 describes the fix-and-relax heuristic for CTA. Finally, Section 4 presents extensive computational results, showing the effectiveness of the approach for some classes of tables.

2. The MILP formulation of the CTA problem

Any CTA problem instance, either with one table or with any number of tables, 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.
- 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, $i \in \mathcal{S}$.

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

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

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

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

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

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

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

3. Fix-and-relax heuristic 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 weeks or days) of execution. When the number of sensitive cells is large, the branch-and-cut scheme has shown to be inefficient, in some cases even 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 linear problems (LPs) obtained by fixing the value of binary variables—associated to the protection senses—are even not solvable with moderate computational resources. For example, the LPs derived from the six million cells instances of the above web address exhaust 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 fix-and-relax heuristic for CTA. Fix-and-relax is a decomposition method based on partitioning the set of binary variables into clusters to iteratively solve a sequence of MILPs of smaller complexity 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 fix-and-relax iteration, a computational improvement is expected. Fix-and-relax can both be seen as an approach for obtaining (hopefully good) initial feasible solutions and primal bounds. There are other approaches for initial good solutions in MILPs, like the feasibility pump [1, 2, 12], but in practice fix-and-relax provided solutions with better optimality gaps (see below Table 5 of Section 4).

Fix-and-relax 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

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

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 fix-and-relax 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 fix-and-relax 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)

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

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 fix-and-relax 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 fix 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 fix-and-relax algorithm for CTA is shown in Figure 3.

4. Computational results

The fix-and-relax heuristic for CTA has been coded in C++, using the state-of-the-art CPLEX 12.4 branch-and-cut solver for the solution of subproblems (4). This approach was compared with the direct solution of (3) through CPLEX branch-and-cut, which will be referred as "plain CPLEX branch-and-cut" or simply "plain branch-and-cut". All the runs were carried out on a

Instance	n	s	m	nz
Random 1H2D instances				
asym-30-40-5-2	76137	3712	4440	154816
asym-30-40-5-5	75657	3689	4428	153857
asym-30-40-5-10	75476	3680	4424	153493
asym-30-40-10-2	75301	4583	4420	153143
asym-30-40-10-5	75451	5149	4423	153445
asym-30-40-10-10	75823	5553	4432	154187
asym-30-50-5-2	78536	5424	4524	159703
asym-30-50-5-5	80610	5328	4593	163917
asym-30-50-5-10	82151	5251	4646	167052
asym-30-50-10-2	83418	5655	4688	169626
asym-30-50-10-5	84420	5982	4722	171665
asym-30-50-10-10	85225	6251	4750	173302
asym-40-40-5-2	86399	6147	4772	175625
asym-40-40-5-5	87550	6065	4794	177909
asym-40-40-5-10	88296	5982	4808	179381
asym-40-40-10-2	89187	6233	4825	181148
asym-40-40-10-5	89779	6436	4835	182318
asym-40-40-10-10	90463	6631	4849	183674
asym-40-50-5-2	92541	6618	4897	187853
asym-40-50-5-5	94398	6604	4940	191587
asym-40-50-5-10	96077	6592	4978	194961
asym-40-50-10-2	97428	6853	5010	197679
asym-40-50-10-5	98768	7102	5041	200374
asym-40-50-10-10	100002	7330	5070	202857
sym-30-40-5	76137	3712	4440	154816
sym-30-40-10	75178	7330	4417	152897
sym-30-50-5	93432	4577	5045	190026
sym-30-50-10	93014	9114	5037	189190
sym-40-40-5	101844	4966	5067	206230
sym-40-40-10	103935	10136	5118	210412
sym-40-50-5	126970	6221	5703	257101
sym-40-50-10	127480	12493	5713	258121
Real instances				
cbs	11163	2467	244	22326
dale	16514	4923	405	33028
destatis	5940	621	1464	18180
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
osorio	10201	7	202	20402
table1	1584	146	510	4752
table3	4992	517	2464	19968
table4	4992	517	2464	19968
table6	1584	146	510	4752
table7	624	17	230	1872
table8	1271	3	72	2542
toy3dsarah	2890	376	1649	9690

Table 1: Characteristics for symmetric/asymmetric random 1H2D and real instances.

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 general and random 1H2D tables. Real general tables are standard instances used in the literature [5]. Random tables were obtained with a generator of 1H2D synthetic 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 random generator is available from `http://www-eio.upc.es/~jcastro/generators_csp.html`. We fixed all parameters, but three: the number of rows per subtable ($\mathtt{r} \in \{30, 40\}$), the number of columns per subtable ($\mathtt{c} \in \{40, 50\}$) and the percentage of sensitive cells ($\mathtt{s} \in \{5, 10\}$). 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, and all the reported results are averaged on these five replications. This amounted to 24 and eight samples of five instances each one, for asymmetric and symmetric instances respectively. Table 1 reports the characteristics of the symmetric/asymmetric 1H2D instances and real tables: the number of cells ("n"), the number of sensitive cells ("s"), the number of table relations ("m") and the number of coefficients in linear constraints ("nz"). Hierarchical tables are identified by the particular combination of parameters, i.e., sym-`r`-`c`-`s` for symmetric instances and asym-`r`-`c`-`s`-`a` for asymmetric ones. 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 the fix-and-relax heuristic mainly 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 fix-and-relax. This empirical analysis was done for the asymmetric 1H2D instances considering values $k \in \mathcal{K} = \{3, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Each instance was solved $|\mathcal{K}|$ times, randomly partitioning the set $\mathcal{S}$ of sensitive cells into $k \in \mathcal{K}$ subsets. The stoping criterion for all the runs, i.e., subproblems (4), was a 5% optimality gap, which is computed by CPLEX as $(\mathrm{UB} - \mathrm{LB})/(|\mathrm{UB}| + 10^{-10})$, 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 4 reports the CPU time (in seconds, averaged for the five replications of each instance) used by fix-and-relax 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 [9]. 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 fix-and-relax 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 fix-and-relax with k clusters over the strategy with the best (minimum) performance for this instance.

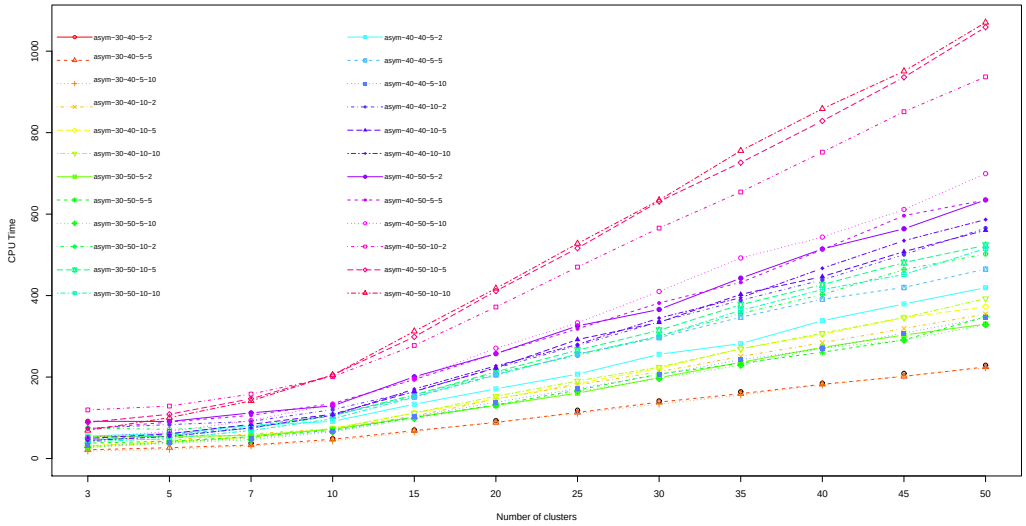

Figure 4: CPU time for different numbers of clusters

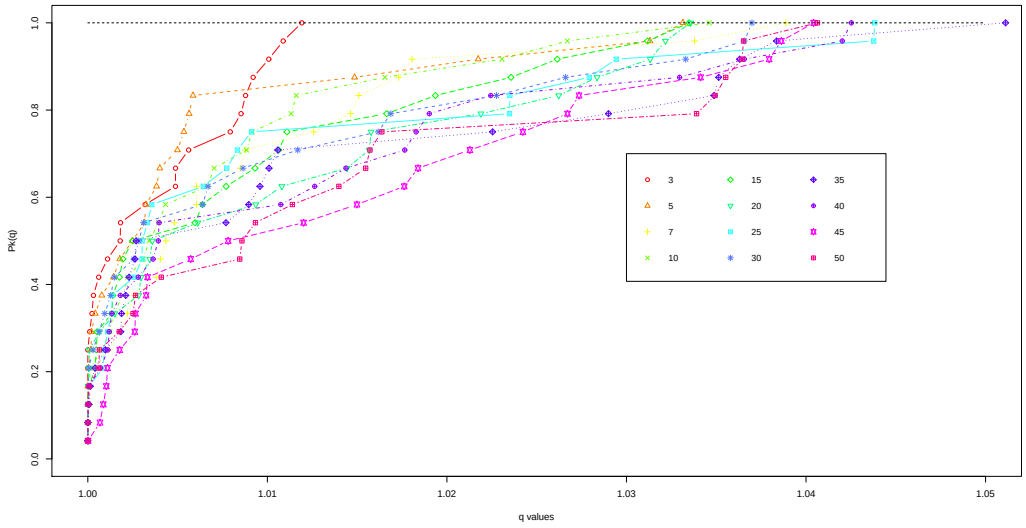

Figure 5: Performance profile of the quality of the solution for different numbers of clusters

	k=3		k=5		k=7		k=10		k=15		k=20		k=25		k=30		k=35		k=40		k=45		k=50	
instance	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap	CPU	pgap
asym-30-40-5-2	38.79	4.93	37.66	5.13	37.68	5.56	48.75	7.03	70.17	7.33	93.19	6.89	118.54	7.25	141.52	6.91	163.82	7.66	184.80	7.47	208.90	6.70	228.93	7.99
asym-30-40-5-5	21.47	1.92	26.57	1.11	33.23	0.78	47.29	1.06	68.60	0.78	88.40	0.75	113.17	0.75	138.57	0.80	159.26	0.75	182.31	0.75	201.64	0.89	224.85	1.17
asym-30-40-5-10	18.15	0.52	22.50	0.51	31.12	0.51	43.53	0.49	65.83	0.49	90.12	0.49	110.78	0.70	133.63	0.49	155.91	0.49	180.72	0.49	202.72	0.49	222.84	0.49
asym-30-40-10-2	50.45	5.19	45.24	7.98	60.41	8.51	73.08	8.18	104.05	8.13	145.97	7.91	180.66	8.96	213.05	8.09	250.52	8.34	284.25	8.84	318.99	8.46	354.86	8.37
asym-30-40-10-5	31.02	2.24	42.62	2.22	51.83	2.16	74.51	2.11	111.66	1.76	146.35	2.06	184.04	1.97	221.40	1.96	269.74	1.98	304.61	2.10	345.82	2.03	372.56	2.57
asym-30-40-10-10	28.78	0.65	39.24	0.49	53.48	0.48	73.05	0.47	112.07	0.53	152.71	0.48	190.47	0.47	224.46	0.47	269.24	0.74	307.56	0.47	346.13	0.56	392.62	0.47
asym-30-50-5-2	54.37	4.50	54.43	6.11	56.30	4.78	71.90	4.89	101.29	4.01	130.23	4.96	160.50	6.77	198.43	5.43	236.23	4.88	272.85	6.65	303.22	5.57	329.78	5.30
asym-30-50-5-5	37.79	2.14	42.60	1.60	53.61	2.28	69.43	1.85	100.62	2.47	131.81	3.26	167.94	1.79	207.35	3.31	234.17	1.70	261.06	3.12	291.11	3.89	348.10	3.25
asym-30-50-5-10	27.23	0.22	37.18	0.38	46.31	0.40	65.40	0.36	98.03	0.14	130.67	0.14	164.81	0.14	196.42	0.33	229.04	0.28	271.21	0.38	290.91	0.29	328.30	0.29
asym-30-50-10-2	74.86	3.66	71.48	6.43	79.70	5.30	107.12	5.33	155.37	5.87	210.21	6.12	253.05	5.91	300.32	6.94	355.95	6.79	402.64	7.28	463.86	7.27	502.27	6.94
asym-30-50-10-5	44.24	3.22	56.79	2.89	74.43	4.32	104.77	3.17	156.43	2.94	212.87	3.10	265.85	2.86	314.73	2.84	377.50	2.77	426.79	2.86	480.93	4.04	523.44	2.79
asym-30-50-10-10	41.11	0.68	48.94	0.45	68.86	0.56	97.49	0.52	150.51	0.57	204.77	0.75	256.44	0.42	300.06	0.47	363.65	0.58	414.77	0.56	451.73	0.50	515.80	0.47
asym-40-40-5-2	55.41	5.09	61.49	4.11	80.93	5.33	91.24	4.79	133.28	4.97	171.04	4.68	207.15	4.65	255.84	4.63	281.95	4.75	338.33	4.86	379.43	4.81	419.34	5.01
asym-40-40-5-5	45.69	1.54	59.06	2.02	76.86	1.99	105.65	1.64	151.85	2.39	206.48	1.77	256.21	1.99	296.84	2.20	346.98	1.68	390.92	1.69	419.94	1.98	465.06	1.90
asym-40-40-5-10	30.79	1.22	40.24	1.21	49.99	1.22	67.39	1.21	102.68	1.40	138.28	1.21	171.53	1.31	206.18	1.21	242.68	1.40	269.86	1.21	308.13	1.40	345.22	1.21
asym-40-40-10-2	74.20	4.32	83.31	4.88	91.11	6.03	120.19	6.41	165.52	6.69	221.63	7.10	277.39	8.10	335.53	7.34	388.32	7.43	438.48	5.35	500.58	5.83	565.79	5.76
asym-40-40-10-5	49.43	1.91	60.73	1.74	83.91	1.45	108.75	1.23	164.45	1.95	222.77	3.38	292.03	2.40	334.20	1.62	402.99	4.74	446.57	1.47	507.89	3.89	560.35	1.50
asym-40-40-10-10	44.00	1.49	54.10	0.83	75.30	1.46	106.32	0.83	170.47	0.81	227.42	0.75	280.00	0.83	344.75	0.76	394.87	0.77	467.04	1.12	534.81	2.06	586.56	1.47
asym-40-50-5-2	91.41	4.49	91.01	5.09	112.05	4.59	129.30	5.09	200.72	5.08	257.32	5.16	325.47	4.76	365.92	4.77	442.76	6.73	514.39	6.22	563.99	7.03	634.78	7.63
asym-40-50-5-5	73.07	1.29	89.47	1.86	106.09	1.92	135.10	2.00	193.94	2.98	257.89	2.50	318.30	1.63	381.70	2.55	432.31	2.19	512.79	2.96	596.16	2.05	634.63	2.16
asym-40-50-5-10	51.59	1.17	66.52	1.17	93.20	1.16	133.28	1.12	195.21	1.12	270.78	0.99	333.05	1.23	410.03	1.13	492.33	1.21	543.63	1.09	611.32	1.03	699.34	0.85
asym-40-50-10-2	119.35	4.94	128.76	3.64	158.19	6.62	200.95	4.82	277.47	5.42	371.90	5.04	469.99	6.01	565.62	5.61	654.26	6.14	752.10	6.66	851.42	7.19	936.90	6.98
asym-40-50-10-5	88.87	3.22	108.66	4.57	146.73	3.51	205.14	3.28	298.74	3.40	411.31	5.54	516.35	2.82	631.05	2.30	726.41	3.10	828.66	4.00	935.91	4.92	1058.86	5.51
asym-40-50-10-10	68.91	0.90	100.46	0.88	141.53	1.29	204.82	1.23	312.30	1.02	417.49	1.16	527.69	1.68	634.22	1.73	755.57	1.89	858.51	1.19	950.67	1.14	1070.46	2.04
Total	**52.54**	**2.68**	**61.21**	**2.96**	**77.61**	**3.19**	**103.51**	**3.04**	**152.55**	**3.19**	**204.65**	**3.39**	**255.89**	**3.36**	**306.32**	**3.28**	**359.43**	**3.53**	**410.61**	**3.52**	**461.09**	**3.76**	**513.40**	**3.67**

Table 2: Fix-and-relax results for asymmetric 1H2D instances and $k \in \mathcal{K}$

The (cumulative) distribution function $P_k(q) : [1, \infty) \rightarrow [0, 1]$ 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 5 shows the performance profiles for the different $k \in \mathcal{K}$. $P_k(q) = 1$ means fix-and-relax 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 $q = 1.01$ (i.e., fix-and-relax with 3 blocks solves all the instances within a factor 1.01 of the best ratio). It can also be observed that $k = 3$ provides the highest quality for 40% of the instances ($P_3(1) \approx 0.4$).

Table 2 confirms the previous results. This table reports the CPU time and *primal gap* (averaged for the five replications) for all the instances and $k \in \mathcal{K}$. The primal gap is defined as (UB − LB)/|UB|, where UB is the best feasible solution and LB is the best known lower bound computed with either fix-and-relax or plain CPLEX branch-and-cut (primal gaps will also be used in tables 3–5). The last row of Table 2 shows averaged values for all the instances. Clearly, $k = 3$ provides the fastest executions and at the same time the lowest gap.

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

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

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

From Table 3 it can be concluded that fix-and-relax is more efficient than plain branch-and-cut for fast good feasible solutions of 1H2D tables. In several runs (marked with ‡) branch-and-cut could not find a better solution than fix-and-relax within the time limit. It is worth noting that for all the 1H2D instances fix-and-relax provided solutions with gaps below 10%. For the real 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 fix-and-relax is not so competitive. It only outperforms plain branch-and-cut in eight of the 19 instances, and they both provide the same gap in two adittional cases ("cbs" and "osorio"). In one of these cases ("toy3dsarah") branch-and-cut could not improve the fix-and-relax solution within the time limit. In the remaining instances branch-and-cut outperformed fix-and-relax.

We tried to warm start plain CPLEX branch-and-cut with the fix-and-relax solution. It could be expected that providing a good incumbent from the beginning would reduce the computational

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC,FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T_{FR},T^{up}_{BC})$
asym-30-40-5-2	38.79	4.93	86.26	81.33	3.02	154.09	115.30
asym-30-40-5-5	21.47	1.92	100	98.08	0.51	56.46	34.99
asym-30-40-5-10	18.15	0.52	100	99.48	0.42	49.52	31.37
asym-30-40-10-2	50.45	5.19	100	94.81	$\ddagger^{(2.39,3)}$	$\ddagger^{(434.48,3)}$	$\ddagger^{(384.02,3)}$
asym-30-40-10-5	31.02	2.24	100	97.76	0.32	138.70	107.68
asym-30-40-10-10	28.78	0.65	100	99.35	0.39	72.48	43.70
asym-30-50-5-2	54.37	4.5	$\dagger^{(100,1)}$	$\dagger^{(95.50,1)}$	1.58	216.45	162.07
asym-30-50-5-5	37.79	2.14	$\dagger^{(100,1)}$	$\dagger^{(97.86,1)}$	1.06	79.02	41.23
asym-30-50-5-10	27.23	0.22	100	99.78	0.08	70.65	43.42
asym-30-50-10-2	74.86	3.66	100	96.34	$\ddagger^{(2.45,1)}$	$\ddagger^{(560.14,1)}$	$\ddagger^{(485.29,1)}$
asym-30-50-10-5	44.24	3.22	100	96.78	1.86	224.29	180.05
asym-30-50-10-10	41.11	0.68	100	99.32	0.27	106.30	65.20
asym-40-40-5-2	55.41	5.09	$\dagger^{(100,1)}$	$\dagger^{(94.91,1)}$	1.08	178.16	122.75
asym-40-40-5-5	45.69	1.54	$\dagger^{(100,1)}$	$\dagger^{(98.46,1)}$	0.76	88.39	42.70
asym-40-40-5-10	30.79	1.22	100	98.78	1.14	79.44	48.65
asym-40-40-10-2	74.2	4.32	100	95.68	$\ddagger^{(2.63,3)}$	$\ddagger^{(365.38,3)}$	$\ddagger^{(291.17,3)}$
asym-40-40-10-5	49.43	1.91	100	98.09	0.59	182.68	133.26
asym-40-40-10-10	44	1.49	100	98.51	$\ddagger^{(0.75,1)}$	$\ddagger^{(101.66,1)}$	$\ddagger^{(57.66,1)}$
asym-40-50-5-2	91.41	4.49	$\dagger^{(100,1)}$	$\dagger^{(95.51,1)}$	2.22	283.61	192.19
asym-40-50-5-5	73.07	1.29	100	98.71	0.52	110.03	36.96
asym-40-50-5-10	51.59	1.17	100	98.83	0.67	91.08	39.49
asym-40-50-10-2	119.35	4.94	100	95.06	$\ddagger^{(3.77,1)}$	$\ddagger^{(1235.97,1)}$	$\ddagger^{(1116.61,1)}$
asym-40-50-10-5	88.87	3.22	$\dagger^{(100,2)}$	$\dagger^{(96.78,2)}$	0.64	335.49	246.62
asym-40-50-10-10	68.91	0.90	$\dagger^{(100,2)}$	$\dagger^{(99.10,2)}$	0.68	151.26	82.35
sym-30-40-5	139.06	5.28	100	94.72	‡(4.67, 2)	$\ddagger^{(194.62,2)}$	$\ddagger^{(55.56,2)}$
sym-30-40-10	224.51	8.94	62.69	53.75	5.73	472.62	248.11
sym-30-50-5	202.66	7.60	82.61	75.01	1.97	463.04	260.38
sym-30-50-10	426.43	6.75	$\dagger^{(34.06,1)}$	$\dagger^{(27.31,1)}$	4.75	757.79	331.36
sym-40-40-5	206.97	5.38	$\dagger^{(78.89,1)}$	$\dagger^{(73.51,1)}$	2.69	487.67	280.71
sym-40-40-10	909.91	5.57	63.85	58.28	$\ddagger^{(5.04,3)}$	$\ddagger^{(2972.69,3)}$	$\ddagger^{(2062.79,3)}$
sym-40-50-5	291.80	7.49	82.10	74.62	4.65	701.59	409.79
sym-40-50-10	976.78	5.55	62.80	57.25	$\ddagger^{(7.62,4)}$	$\ddagger^{(226.62,4)}$	$\ddagger^{(-750.16,4)}$

$\dagger^{(x,y)}$ plain branch-and-cut could not find a solution in y of the five replications within T_{FR} seconds;
x is the average value for the $5-y$ successful runs.
$\ddagger^{(z,w)}$ plain branch-and-cut could not improve the fix-and-relax solution in w of the five replications within the time limit;
z is the average value for the $5-w$ successful runs.

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

instance	T_{FR}	$GAP_{FR}\%$	$GAP_{BC}\%$	$\Delta(BC,FR)$	$GAP^{up}_{BC}\%$	T^{up}_{BC}	$\Delta(T_{FR},T^{up}_{BC})$
cbs	2.31	100	100	0	0	2.41	0.1
dale	319.09	88.8	0	−88.8	0	100.72	−218.37
destatis	162.66	16.24	99.97	83.73	2.27	505.09	342.43
hier13	509.65	6.93	4.95	−1.98	5.34	234.76	−274.89
hier13x13x13a	378.49	5.25	4.96	−0.29	4.96	369.96	−8.53
hier13x13x13b	436.29	5.66	7.74	2.08	5.11	961.68	525.39
hier13x13x13c	431.12	5.6	5.09	−0.51	5.09	229.81	−201.31
hier13x13x13d	123.15	7.07	28.06	20.99	4.84	145.49	22.34
hier13x13x13e	272.59	5.43	3.78	−1.65	3.78	152.01	−120.58
hier13x13x7d	37.19	5.15	4.91	−0.24	4.91	33.85	−3.34
hier13x7x7d	3.25	5.26	9.52	4.26	4.97	9.26	6.01
osorio	1.6	0	0	0	0	0.61	−0.99
table1	0.57	8.47	44.93	36.46	4.62	1.29	0.72
table3	810.07	19.11	8.32	−10.79	12.37	368.46	−441.61
table4	450.51	20.11	100	79.89	16.6	473.1	22.59
table6	0.45	11.03	11.42	0.39	9.47	0.78	0.33
table7	0.12	0.56	0.41	−0.15	0.41	0.02	−0.1
table8	0.17	2.44	0	−2.44	1.35	0.03	−0.14
toy3dsarah	21.54	0.36	2.44	2.08	†	†	†

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

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

instance	$GAP_{FR}\%$	T_{FR}	$GAP_{FP}\%$	T_{FP}	$\Delta(T_{FP}, T_{FR})$	$\Delta(FP, FR)$
asym-30-40-5-2	4.93	38.79	69.75	167.20	128.41	64.82
asym-30-40-5-5	1.92	21.47	84.91	173.20	151.73	82.99
asym-30-40-5-10	0.52	18.15	90.40	197.00	178.85	89.88
asym-30-40-10-2	5.19	50.45	72.64	249.40	198.95	67.46
asym-30-40-10-5	2.24	31.02	84.18	281.40	250.38	81.93
asym-30-40-10-10	0.65	28.78	93.07	346.20	317.42	92.42
asym-30-50-5-2	4.50	54.37	63.81	221.20	166.83	59.31
asym-30-50-5-5	2.14	37.79	80.92	242.80	205.01	78.78
asym-30-50-5-10	0.22	27.23	93.60	308.80	281.57	93.38
asym-30-50-10-2	3.66	74.86	68.60	372.40	297.54	64.94
asym-30-50-10-5	3.22	44.24	84.32	401.80	357.56	81.10
asym-30-50-10-10	0.68	41.11	92.56	495.60	454.49	91.89
asym-40-40-5-2	5.09	55.41	66.97	231.60	176.19	61.88
asym-40-40-5-5	1.54	45.69	83.46	256.60	210.91	81.92
asym-40-40-5-10	1.22	30.79	92.28	287.60	256.81	91.06
asym-40-40-10-2	4.32	74.20	65.99	368.20	294.00	61.67
asym-40-40-10-5	1.91	49.43	87.95	467.80	418.37	86.04
asym-40-40-10-10	1.49	44.00	97.45	734.60	690.6	95.96
asym-40-50-5-2	4.49	91.41	72.74	313.20	221.79	68.24
asym-40-50-5-5	1.29	73.07	82.17	350.80	277.73	80.88
asym-40-50-5-10	1.17	51.59	94.78	473.00	421.41	93.61
asym-40-50-10-2	4.94	119.35	70.02	497.00	377.65	65.08
asym- 40-50-10-5	3.22	88.87	82.56	546.00	457.13	79.34
asym-40-50-10-10	0.90	68.91	92.80	718.00	649.09	91.89
sym-30-40-5	5.28	139.06	45.57	150.6	11.54	40.29
sym-30-40-10	8.94	224.51	44.57	189.2	−35.31	35.63
sym-30-50-5	7.6	202.66	58.52	216.6	13.94	50.92
sym-30-50-10	6.75	426.43	65.02	353.2	−73.23	58.27
sym-40-40-5	5.38	206.97	60.7	218.6	11.63	55.32
sym-40-40-10	5.57	909.91	55.12	366.8	−543.11	49.55
sym-40-50-5	7.49	291.8	52.6	305.6	13.8	45.12
sym-40-50-10	5.55	976.78	52.45	458.6	−518.18	46.9

Table 5: Comparison between fix-and-relax and feasibility pump heuristic for symmetric and asymmetric random 1H2D instances.

burden by pruning portions of the search space. However, the results were not satisfactory, but rather disapointing. In fact, we found reported similar experiences. In `http://www2.isye.gatech.edu/~rcarvajal3/2` the author presents an experiment with instances from MIPLIB 2010 [17] where providing the optimal solution as a warm start can actually be harmful for the performance of the solver.

As a last experiment, we compared fix-and-relax with feasibility pump [12], both seen as efficient heuristics for the fast computation of hopefully good initial feasible solutions to MILPs. We used the "objective feasibility pump" variant [1], which is more efficient in terms of quality of the solution. Table 5 reports the primal gap of the fix-and-relax and feasibility pump solutions (columns "$GAP_{FR}\%$" and "$GAP_{FP}\%$", respectively), the CPU time required by fix-and-relax and feasibility pump to compute the feasible solution (columns "T_{FR}" and "T_{FP}", respectively), and the difference between feasibility pump and fix-and-relax CPU times and gaps (columns "$\Delta(T_{FP}, T_{FR})$" and "$\Delta(FP, FR)$", respectively). It is clearly seen that fix-and-relax outperformed feasibility pump for CTA, both in terms of efficiency and quality of the solution. Fix-and-relax always provided a better gap than feasibility pump, and in most cases by a big difference. Moreover, fix-and-relax was also much more efficient than feasibility pump for all the asymmetric instances, and for all but four symmetric cases. In this four cases, however, the gap provided by fix-and-relax was much smaller. It can be concluded that, for the CTA problem, fix-and-relax instead of feasibility pump should be used for fast and good feasible solutions.

5. Conclusions

Fix-and-relax has shown to be an efficient heuristic for the difficult MILP CTA problem. Initially developed for scheduling problems that can be partitioned into stages, fix-and-relax has also been successfully applied to a class of hierarchical tables named 1H2D. For these tables, it was competitive against both plain CPLEX branch-and-cut and the feasibility pump heuristic. For general tables, fix-and-relax also outperformed the other approaches in half of the cases. 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. Fix-and-relax is thus an step in this direction. Combining fix-and-relax with other heuristics, such as block-coordinate-descent, or embedding fix-and-relax 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 grants MTM2012-31440 of the Spanish Government and SGR-2009-1122 of the Government of Catalonia.

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] 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).
[9] E.D. Dolan, J.J.Moré, Benchmarking optimization software with performance profiles, Mathematical Programming, A, 91, 201–13 (2002).
[10] 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).
[11] 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).
[12] M. Fischetti, F. Glover, A. Lodi, The Feasibility Pump, Mathematical Programming, 104, 91–104 (2005).
[13] M. Fischetti, J.J. Salazar. Solving the cell suppression problem on tabular data with linear constraints, Management Science, 47, 1008–1026 (2001).
[14] 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).
[15] J.A. González, J. Castro, A heuristic block coordinate descent approach for controlled tabular adjustment, Computers & Operations Research, 38, 1826–1835 (2011).
[16] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. Schulte Nordholt, K. Spicer, P.-P. De Wolf, Statistical Disclosure Control. Chichester, Wiley, 2012.
[17] T. Koch, et al. MIPLIB 2010. Mixed integer programming library version 5, Mathematical Programming Computation, 3, 103–163 (2011).
[18] L. Zayatz, U.S. Census Bureau, communication at Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality, Bilbao (Basque Country, Spain) (2009).