CASD-TERALAB, A SECURE REMOTE ACCESS SYSTEM TO CONFIDENTIAL BIG DATA:

TABULA DATA ANONYMIZATION USING BIG DATA TOOLS.

Said KHADRAOUI: DATA SCIENTIST (TeraLab/CASD)

https://casd.eu/

Keywords: Big Data, data science, anonymization, analytics, remote access, security, privacy, tabu search.

Introduction

Usually, individual data are obtained under a pledge of confidentiality; statistical entities aim at publishing as much information as possible without compromising the identities of their respondents. This results into a trade-off between privacy protection and information loss. In order to avoid data disclosure, cell suppression is a widely used technique. This approach consists first in an identification of sensitive cells that should be suppressed (primary suppression), then a phase of additional cells suppression (secondary suppression) due to margins publication, to ensure table protection with the specified levels of protection.

At TeraLab and INSEE (French National Institute for Statistics and Economic Studies), tau-Argus[1] have been used to deal with table confidentiality, but the software fails when it comes to dealing with k-dimensional tables with $k > 5$. To confront these limitations, an alternative way consists of successive calls to tau-Argus as performed by a SAS Macro proposed by Destatis (the German Federal Statistical Office). The bottleneck of these techniques, inter alia, is the difficulty to deal with high dimensional tables (typically more than 5 explanatory variables), since all these entries are linked by a generic system of linear constraints that takes an exponential time (NP-hard) to be solved. Moreover, even with low dimensions, as soon as the quantity of data contained in the table is voluminous, traditional systems can't perform as expected, this is why we focused, within TeraLab, on a solution with Big Data tools.

Our work addresses this problem with a general configuration, in which we cover **k**-dimensional (**k >5**) tables with margins, as well as hierarchical and linked tables within a big data ecosystem. In particular, we address the optimization problem known in the scientific literature as the (complementary or secondary) cell suppression problem, in which the information loss owing to suppression must be minimized. To our knowledge, problems of this kind have never been solved optimally by previous projects. Moreover, we have also implemented two Impala (Which is more memory bound making it multitude faster than disk based MapReduce tools like Hive) queries that allowed us to carry out the first two steps of tau-Argus process (Primary suppression and tabulation). We explored a new integer linear programming model and then implemented a meta-heuristic algorithm in order to have a reasonable solution for that model. The solution we proposed is based on the use of the technique called Tabu Search that guides a local heuristic search procedure to explore the solution space beyond local optimality thanks to its use of adaptive memory in order to find a near-optimal solution for this problem. In our case, we applied TS to search the optimal cell suppression. This goal is performed through the resolution of the mixed integer linear programming (MILP)[2] that express efficiently the secondary cell suppression model.

Tabular Data Anonymization Process

Data

We used ADSD (Annual Declaration of Social Data) which is a dataset that contains several pieces of information that are transmitted to FISC administrations. We choose to work on this category of data because of the fact that the software tau-Argus that, INSEE utilize for its confidentiality needs, is usually used to anonymize this category of data. These data are composed from a lot of explanatory variables that can be used by all sort of attackers to identify concerned individual.
The response variable that we used to anonymize our data is individual salary, it consists of the monthly remuneration perceived by individuals and given by their employer.

Tau-Argus

They are three principal ways to deal with a secondary cell suppression problem in tau-Argus.
The Hypercube algorithm gives a fast solution that may cause over-protection. The Optimal method solves a problem of Mixed Integer Linear Programming and the Modular approach enables to find more quickly a near optimal solution for hierarchical tables.
The Hypercube method synthesized in Giessing and Riepsilber (2002)[3] is a heuristic that finds very quickly a solution even for very detailed and high-dimensional tables. After having subdivided the initial table into sub-tables without substructure, each primary unsafe cell is protected successively by finding the best suppression pattern in the form of an hypercube: a sensitive cell is sufficiently protected if it is contained in an hypercube where the corner points are suppressed cells. This method often leads to over-protection because the set of suppression patterns considered is restricted. This method bugs when it comes to solve **k**-dimensional tables (**k > 5**).

Tabulation

The very first step in the cell suppression process on tabular data is to construct the table to be processed from micro data. This step, compared to the steps of the application of the primary and secondary secret is easier to do since it does not present algorithmic difficulties and does not require significant computing powers. It is simply a matter of constructing an array with the selected explanatory variables and including the hierarchies of these variables. An explanatory variable can have a hierarchical structure at several levels. As an example, a variable of the address of a given place in France. We go up the hierarchy to reach the town, then the county and finally the region.
The goal is to aggregate the response variable on the different values of the specified explanatory variables. The table produced is a table in which the value of each cell represents the sum of the contributions of respondents falling into that cell. It is primarily a matter of specifying the explanatory variables and the response variable that will be used to construct the table to be anonymized. The explanatory variables can have hierarchical structures that must be taken into account in the construction of the table. It follows a step of aggregating the response variable on the different explanatory variables and the different levels of their hierarchies.
Tau-Argus reaches its limits during the tabulation stage and this for an ADSD file corresponding to a single county from 5 explanatory variables. Since tabulation is the most basic step in the process of anonymization by suppression, it seems meaningful for us to move towards another alternative.
Within TeraLab, tests have been conducted to define the limits of tau-Argus software on a physical machine with sufficient GB of available RAM. Initially, the software was run on the 2012 ADSD data of a single county with a number of reduced explanatory variables that are incrementally increased to see where the boundary is. Once the number of variables is increased, the tabulation and anonymization

time takes longer and longer.
The table below gives an idea of the performance of tau-Argus on a file of 1,306,620 rows (37.3 MB), on a dedicated physical machine:

Number of explanatory variabless	Secret secondaire (seconds) (« modular »)
4	3h27
>5	Can't work (bugs)

We see that the limit of this software is reached as soon as the number of explanatory variables grows, typically, when we exceed 5 variables, tau-Argus bugs and can't give any result.

These results confirm the hypothesis of the exponential processing time as a function of the number of variables and confirm the solution to turn to other alternatives for problem solving based on Big Data tools.

Different use cases

These "simple" examples help to better understand the approach described above.

1. Without hierarchy

SAS original data: p explanatory variables $V_{b,1}$, …. $V_{b,p}$ and R the response variable.
The tabulation creates an array of p* explanatory variables V_1, … V_p where $p^* \leq p$.

2. A single explanatory variable with m hierarchy levels
    a. p* =1, m=2:

V has a hierarchical structure with two levels: V^1 and V^2 and V^1 is the finest level and V^2 is the most aggregated.

Example of micro-data:

V	**V^1**	**V^2**
2	0	0
2	0	0
10	2	0
9	1	1
9	1	1
9	1	1
15	3	1

The searched table:

V	**V^1**	**V^2**	**sum**
2	0	0	2
10	2	0	1
9	1	1	3
15	3	1	1
NULL	0	0	2
NULL	2	0	1
NULL	1	1	3
NULL	3	1	1
NULL	NULL	0	3

NULL	NULL	1	4
NULL	NULL	NULL	7

b. p* =1, m any integer:

V, V^1,…, V^m where V^1 is the finest level and V^m is the most aggregated one.

The table we want to obtain has the following form where x denotes values and NULL denotes an empty field:

V	**V^1**	**…**	**V^m**	**sum**
x ⋮ x	x ⋮ X		x ⋮ x	x ⋮ x
NULL ⋮ NULL	x ⋮ X		x ⋮ x	x ⋮ x
NULL ⋮ NULL	NULL ⋮ NULL		x ⋮ x	x ⋮ x
NULL	NULL	NULL	NULL	x

c. Three explanatory variables, a single one with hierarchy

(1) (2) (3) (4)

dads_idf_micro ➔ dads_idf _micro_7 ➔ dads_idf_micro_clean ➔ dads_idf_micro_h ➔

(5)

dads_idf_tab ➔ dads_prim

Table	**Rows number**
dads_idf_micro	58 825 580
dads_idf_micro_clean	19 012 113
dads_idf_micro_7_h	19 459 448

Primary cell suppression

Once the aggregate table is constructed, to ensure primary secret, it is necessary to traverse all the cells of the table and to hide the cells which do not satisfy at least one of the following two criteria:

- Frequency rule: a cell must be constructed from at least k units (at INSEE ***k = 3***)
- Rule of dominance: the cell first contributor must not represent more than ***x%*** of the total value of the cell (to INSEE ***x = 85%***)

Two rules described in Nicolas (2010)[4] are applied to protect French business statistics: a three-unit rule and the dominance (85%) rule. Let a cell $T_c = \sum_{i=1}^{n} w_i\, x_i$ where$x_1 \geq \cdots . \geq x_n$. x_i denotes the response value and w_i its associated weight. The cell is considered sensitive if $w_i \leq 3$ (frequency rule) or $x_1 > 0.85 \times T_c$ (dominance rule).

For the three methods tested in tau-Argus, additional protection against singleton disclosure is provided. Additivity and non-negativity constraints of the published table enable an attacker to derive estimates from an unpublished cell. tau-Argus computes protection intervals to ensure that a sensitive cell cannot be estimated too precisely. We consider the following protection intervals:

$[0.9 \times T_c, 1.1 \times T_c]$ for a frequency problem.

$[2 \times T_c - \frac{x_1}{0.85}, \frac{x_1}{0.85}]$ for a dominance problem.

Note that if a cell is sensitive because of the two sensitivity rules, e.g. a cell with a unique contributor, protection intervals are computed as if it is a dominance problem. The protection is made with respect to protection intervals and the cost function is equal to the cell value for all three methods.

Secondary Cell Suppression

Here we give a formal definition of the CSP that we address in this article. A table is a data vector ***a = [a_1,...,a_n]*** whose entries satisfy a given set of linear constraints known to a possible attacker,

$$\mathbf{My = b}$$

$$\mathbf{lb_i \le y_i \le ub_i} \quad \forall i = 1 \ldots\ldots\ldots n \quad (1)$$

In other words, (1) models the whole a priori information on the table known to an attacker. Typically, each equation in (1) corresponds to a marginal entry, whereas inequalities enforce the "external bounds" known to the attacker. In the case of k-dimensional tables with marginals, each equation in (1) is of the type $\sum_{j \in Qi} y_i - y_j$, where index ***i*** corresponds to a marginal entry and index set ***Q_i*** to the associated internal table entries. Therefore, in this case ***M*** is a ***{0, ±1}*** matrix and ***b = 0***. Moreover, in case ***k = 2***, the linear system (1) can be represented in a natural way as a network, a property with important theoretical and practical implications.

Unfortunately, this nice structure is not preserved for ***k ≥ 3***, unless the table decomposes into a set of independent two-dimensional sub-tables. A cell is an index corresponding to an entry of the table.

Given a nominal table ***a***, let ***PS*** = $\{i_1, \ldots, i_p\}$ be the set of sensitive cells to be protected, as identified by the statistical office according to some criteria. For each sensitive cell $i_k \quad (k = 1, \ldots, p)$, the statistical office provides three nonnegative values: ***LPL_k, UPL_k,*** and ***SPL_k***, the lower protection level, upper protection level, and sliding protection level, whose roles are discussed later. In typical applications, these values are computed as a certain percentage of the nominal value a_{i_k}.

A suppression pattern is a subset of cells, ***SUP ⊆ { 1, ... , n},*** corresponding to the unpublished cells. A consistent table with respect to a given suppression pattern SUP and to a given nominal table is a vector ***y = [y_1, . . . , y_n]*** satisfying:

$$\mathbf{My = b}$$

$$\mathbf{lb_i \le y_i \le ub_i} \quad \forall i \in SUP \quad (2)$$

$$y_i = a_i \qquad \forall i \notin SUP$$

where the latter equations impose that the components of y associated with the published entries coincide with the nominal ones. In other words, any consistent table gives a feasible way for the attacker to fill in the missing entries of the published table.

A suppression pattern is considered feasible by the statistical office if it guarantees the required protection intervals against an attacker, in the sense that for each sensitive cell $i_k \, (k = 1, \ldots, p)$ there exist two feasible tables, say f^k and g^k, such that $f_{i_k}^k \le a_{i_k} - LPL_K$, $g_{i_k}^k \ge a_{i_k} + UPL_k$, and $g_{i_k}^k - f_{i_k}^k \ge SPL_k$.

The Integer Linear programming model

For notational convenience, we define the relative external bounds,

$$LB_i := a_i - lb_i \geq 0$$

and

$$UB_i := ub_i - a_i \geq 0$$

that is, the range of feasible values for cell i known to the attacker is $[a_i - LB_i; a_i + UB_i]$. To obtain a mixed integer linear programming (MILP) model for the CSP, we introduce a binary variable $x_i = 1$ for each cell i, where $x_i = 1$ if $i \in$ SUP (suppressed), and $x_i = 0$ otherwise (published). Clearly, we can fix $x_i = 0$ for all cells that must be published (if any) and $a_i = 1$ for all cells that must be suppressed (sensitive cells). Using this set of variables, the model is of the form

$$min \sum_1^n w_i x_i \qquad (3)$$

subject to $x \in \{0,1\}^n$ and, for each sensitive cell i_k $(\boldsymbol{k} = \boldsymbol{1}, \dots, \boldsymbol{p})$

The suppression pattern associated with x satisfies the upper, lower, and sliding protection-level requirements with respect to cell i_k (4)

The model

We next propose a new model inspired by Benders' decomposition (see, e.g., Nemhauser and Wolsey 1988). The idea is to use standard LP duality theory to project the auxiliary variables f^k and g^k ($k = 1,\dots, p$) away from the model. In the new formulation, the protection-level requirements are in fact imposed by means of certain families of linear constraints in the space of the x-variables only.
The new model is based on a characterization (reported in the Appendix) of the vectors x that minimizes (3) so that the new model can be summarized to:

$$min \sum_1^n w_i x_i$$

subject to $x \in \{0,1\}^n$ and, for each sensitive cell i_k $(\boldsymbol{k} = \boldsymbol{1}, \dots, \boldsymbol{p})$

$$\sum_1^n (\alpha_i UB_i + \beta_i LB_i) x_i \geq UPL_k$$
$$\forall\ (\alpha, \beta, \gamma) satisfiying\ (A.1) \qquad (5)$$

$$\sum_1^n (\alpha_i UB_i + \beta_i LB_i) x_i \geq LPL_k$$
$$\forall\ (\alpha, \beta, \gamma) satisfiying\ (A.2) \qquad (6)$$

and

$$\sum_1^n [(\alpha_i + \alpha_i')UB_i + (\beta_i + \beta_i')LB_i] x_i \geq UPL_k$$
$$\forall\ (\alpha, \beta, \gamma) satisfiying\ (A.1)$$
$$\forall\ (\alpha', \beta', \gamma') satisfiying\ (A.2) \qquad (7)$$

where (A.1) and (A.2) are linear systems governing the projection, as defined in the Appendix. Although these systems admit in general an infinite number of feasible solutions (α, β, γ) satisfying (A.1) and $(\alpha', \beta', \gamma')$ satisfying (A.2), it is well known that only a finite number of "extreme" solutions lead to undominated constraints (5)-(7); that is, these constraints are finitely many and hence define a convex polyhedron. We call (5)-(7) the capacity constraints, in analogy with similar constraints introduced by Fischetti and Salazar (1999) for enforcing a sufficient "capacity" of certain cuts in the network representation of the CSP on two-dimensional tables with marginals. Intuitively, the capacity constraints force suppression (i.e., to set $x_i = 1$) of a sufficient number of cells whose positions within the table and contributions to the overall protection are specified by the "dual variables" (α, β, γ) of the attacker subproblems defined in the Appendix.

Solving the model

Tabu-Search

The model described in the previous section was resolved by implementing a meta-heuristic optimization algorithm which called tabu-search, the algorithm aims at finding the optimal solution of the LP problem. Tabu-Search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. One of the main components of Tabu-Search is its

use of adaptive memory, which creates a more flexible search behavior. Memory-based strategies are therefore the hallmark of tabu search approaches, founded on a quest for "integrating principles," by which alternative forms of memory are appropriately combined with effective strategies for exploiting them. As we begun in this experimentation, we introduced a simple tabu-search, based only on short term memory (Algorithm1).

This procedure has one parameter: the number of iterations, ***N***, which is used as the stopping criterion. The other arguments are a seed for initializing the random points chosen for cell suppression, and the name of the data file of dads.

The part of the MILP solution that is determined by tabu-search is the subset of integer variables ***x*** in Equation 3. The data structure representing a solution is therefore an ***n***-dimensional vector of integers, $\boldsymbol{X} = (x_1, \dots, x_n)$, ***n*** is the number of cells to be suppressed, it is therefore variable and can change through the progress of the tabu search.

Algorithm 1: Simple tabu search

```
Tabu(N, seed, DATA)
1       Read global tabular data from DATA which is stored on HIVE.
2       Initialize random number generator with seed
3       X := CONSTURCT()
4       X* := X
5       t := (-n,…,-n)
6       for i=1 to N
7               X := TABU_MOVE(X,X*,i, t)
8               If x is better then X*
9                       X* := X
        Return X*
```

Construction is based on the solution of the MILP relaxation with a set of variables ***F*** fixed, as stated in equations 5-7 but without the constraint that $\boldsymbol{X}$ should be an integer (empty ***F*** leads to the LP relaxation of the initial problem).

The Solution of this problem is denoted by $\boldsymbol{X^r} = (x_1^r, \dots, x_n^r)$. These values are rounded up or down with some probabilities and fixed (**functions roundup(), rounddown()**), as shown in Algorithm 2, were we denote by $\boldsymbol{v}$ a continuous random variable with uniform distribution within $[0,1]$.

Algorithm 2: Semi-greedy solution construction.

```
CONSTRUCT()
1       F := {}
2       C := {1,…,n}
3       For j=1 to n
4               Solve equations (4-7) with X are in real positive numbers.
5               Randomly select i from C
6               If v < x_i^r − rounddown(x_i^r)
7                       x_i := roundup(x_i^r)
8               Else
9                       x_i := rounddown(x_i^r)
10              F := F ∪ {i}
11              C := C \ {i}
        Return X
```

This semi-greedy construction is inspired in an algorithm provided in [5]. It consists of rounding Each variable ***i*** to an Integer next to its value On the MILP relaxation $\boldsymbol{x_i^r}$.

For all the Indices $\boldsymbol{i} \in \{\boldsymbol{1}, \dots, \boldsymbol{n}\}$, the variable $\boldsymbol{x_i}$ is equal to the value $\boldsymbol{x_i^r}$ rounded down with probability

$$P\big(x_i = rounddown(x_i^r)\big) = roundup(x_i^r) - x_i^r$$

or rounded up with probability $1 - P(x_i = rounddown(x_i^r))$ (lines 6 to 9 of the Algorithm 2)

At each tabu search iteration, the neighborhood of the current solution is searched and a neighbor solution is selected, as presented in Algorithm 3.

The arguments of this algorithm are the current solution ***X***, the best solution found ***X****, the current iteration ***i***, and the tabu vector ***t***. Tabu information is kept in vector ***t;tc*** holds the iteration at which variable ***c*** has been updated.

Algorithm 3: Search of a candidate at each tabu search iteration

```
TABU_MOVE(X, X*,k,t)
1       If k − t_i > n  ∀ i
2               c := R[1,n]
3               x_c := R[l_c, u_c]
4               t_c := k
5               Return X
6       v := X
7       For i = 1 to n
8               s = X
9               d := R[1,n]
10              Foreach δ ∈ {−1; 1}
11                      s_i := x_i + δ
12                      If s_i ∈ [l_i, u_i] and s is better than v
13                              If k − t_i > d or s is better than X*
14                                      v := s
15                                      c := i
16      X := v
```

Lines 1 to 5 prevent the case where the search is blocked, all moves being potentially tabu. In this case a random move is taken: an index is selected randomly, and a value for that variable is drawn within its bounds, with uniform distribution. (We denote by ***R[1;n]*** a random integer with uniform distribution within ***[1,n]***.) The neighborhood is searched by adding a value **= ±**1 to each of the variables ***1,…, n***, as long as they are kept within their bounds. We have tested two search possibilities: breadth first and depth first. With breadth first all the neighbor solutions are checked, and the best is returned (lines 7 to 15).

With depth first, as soon as a non-tabu solution better than the current solution is found, it is returned. Results obtained for these two strategies are rather similar, but there seems to be a slight advantage to breadth first, which is the Strategy that adopted is this paper.

More sophisticated ways of managing choice rules by means of candidate list strategies are an important topic in tabu-search (see, e.g., [6]), and may offer improvements, but we elected to keep this Aspect of the method at a simple level. The tabu tenure (the number of iterations during which an changed variable remains tabu) is generally a parameter of tabu search.

In order to simplify the parameterization, we decided to consider it a random value between 1 and the number of integer variables, ***n***. Such value, ***d***, is drawn independently for each variable (line 9); this might additionally lead to different search paths when escaping the same local optimum, in case this situation arises.

Cluster architecture

Within TeraLab, we have mounted a specific Hadoop cluster on which we have already installed Spark, the cluster is composed from 6 Ubuntu machines. Two machines play the role of master server which

are used to manage and configure the cluster environment, the others four servers are used as worker server; they are specifically boosted in terms of performance in order to support the exigencies of our algorithm.

The Technical specifications of the cluster are listed below:

- 108 GB of RAM
- 18 cores for computational processors (Dell PowerEdge R630).
- 8 TB of disk storage.
- 10 Giga Bytes for Ethernet network.

This architecture was conceived in order to satisfy computational needs that traditional software like tau-Argus couldn't answer.

Benchmark

The implementation of the algorithm above was done by using PySpark and Hive database for storing ADSD, we have used seven explanatory variables which are described on the table below, the response variable was salary (salaire):

Explanatory variables:

sexe (Bigint)	The gender of the individual (1 =M, 0=F).
depr (int)	The number of department where lives tshe individual.
cs (int)	The social category of the individual.
ce (string)	The employment conditions of the individual.
trage (int)	The age of the individual.
treffen (bigint)	The company internal category to which belong the individual.
a88 (int)	The principal activity of the company where the individual works.

Response variable:

salaire (int)	The wage perceived by the individual.

To query on hive tables, we have used impala which is a cloudera's open source massively parallel processing (MPP) SQL query engine for data stored in a computer cluster running Apache Hadoop.

The impala queries were implemented to perform the tabulation and primary secret phases, solving the seven-dimensional instance appears to be much more difficult using traditional optimization methods than meta-heuristics. This is due, of course, to the large number of table links (equations) to be considered. In addition, the number of nonzero protection levels after preprocessing is significantly larger than for the other instances. This results in a large number of time-consuming attacker sub-problems that need to be solved for capacity cut separation, and a large number of capacity cut constraints to be inserted explicitly into the MILP. Moreover, we have observed that the optimal solutions of the MILP tend to have more fractional components with respect to the case of two-dimensional tables with about the same number of cells. In other words, increasing the table dimension seems to have a much greater impact on the number of fractionalities than just increasing the size of a table. Consequently, seven-dimensional tables tend to require more branchings to enforce integrality of the variables. In addition, the tabu-search approache become much more time-consuming, as they work explicitly with all of the nonzero variables of the current fractional MILP

solution. The use of a cluster architecture with parallel processing using PySpark seems to reduce consequently the time consumed during the secondary cell suppression algorithm.

Another issue which appeared to influence very negatively the time used by the algorithm to accomplish the secondary suppression, it is the file format of hdfs storage.

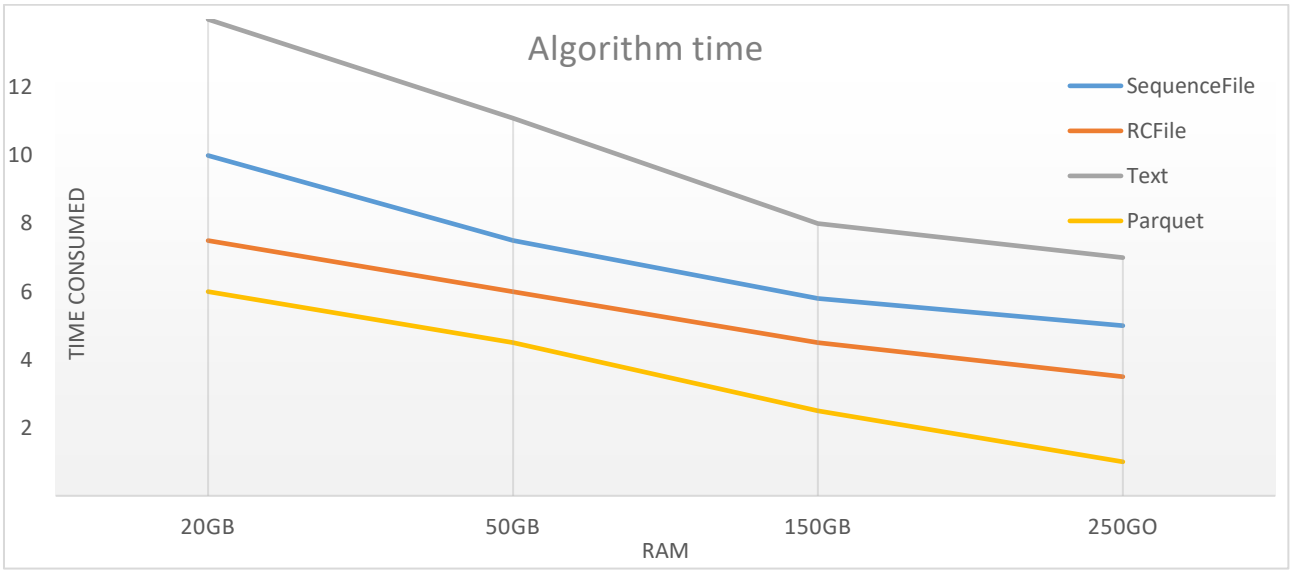
A file format refers to the way in which information is stored or encoded in a computer file. Hadoop offers many ways to store data.

In this section, we will compare three major file formats in Hadoop and the impact they have on the algorithm performances:

- SequenceFile format: a binary storage format for key/value pairs.
- RCFile (Record Columnar File): a binary file format designed for systems using the MapReduce framework.
- Parquet: a column-oriented binary file format which is especially good for queries scanning particular columns within a table.

It's important to note that enabling table's compression provides performance gains in most cases. File compression brings two major benefits: it reduces the space needed to store files, and it speeds up data transfer across the network or to or from disk[7] which is very useful for tabu-search method.

Snappy is recommended for its effective trade-off between compression ratio and decompression speed [8].Furthermore it is supported for the three formats tables.



The chart shows the amount of RAM space used. The job was configured so that the four formats would utilize Snappy compression codec.

It seems straightforward that Parquet format has the best performances compared to RCFile format, SequenceFile format and obviously Text format. Therefore, it will become the chosen standard for tabu-search implementation.

Conclusion

In statistical disclosure control, the problem of anonymization of tabular data can be resolved by the cell suppression methodology. In this paper, in the general context of multi-dimensional tabular data whose entries are linked by a system of linear constraints, we have proposed a new implementation

of a new integer linear programming model for the cell suppression problem (CSP) using big data tools in order to overcome the bugs encountered when tau-Argus is used.
The algorithm used covers multi-dimensional tables with marginals as well as hierarchical and linked tables. We have also outlined a possible solution procedure based on the use of a meta-heuristic approach called tabu-search. This is an initial solution that can be pursued in order to develop a more sophisticated tool based on meta-heuristics approaches.
Since the launch of TeraLab project in 2014, the platform has hosted several projects and Proof of Concepts from both academic and industrial sectors. Some of those projects already went into production after conclusive results on the platform.
The advantage of using TeraLab in dealing with this project is to have more sufficient memory and disk resources in order to satisfy the needs of numerical computations. TeraLab has on its disposal the ability to perform tasks with a lot of use of cluster resources, it allows users to fulfill sophisticated and heavy calculus within a big data architecture.
TeraLab still seeks to strengthen its presence and activity within the French statistical ecosystem as well as on the European level, this is why we consider that the approach of tabular data anonymization we proposed here may draw attention of EUROSTAT and other European statistical institutes.

Appendix

Consider the linear program,

$$\bar{y}_{i_k} \coloneqq \max y_{i_k}, \qquad \textbf{(B.1)}$$

s.t

$$My = b, \qquad \textbf{(B.2)}$$

$$y_i \leq a_i + UB_i x_i \qquad \forall i = 1, \dots., n \qquad \textbf{(B.3)}$$

And

$$-y_i \leq -a_i + LB_i x_i \qquad \forall i = 1, \dots., n \qquad \textbf{(B.4)}$$

This is an LP problem in the y variables only, with variable upper/lower bounds depending on the given parameter x. We call (B. l)-(B.4) the attacker sub-problem associated with the upper protection of sensitive cell i_k, with respect to the given parameter x.
By LP duality, this sub-problem is equivalent to the linear program:

$$\bar{y}_{i_k} = \min \gamma^t b + \sum_{i=1}^{n} [\alpha_i (a_i + UB_i x_i) - \beta_i (a_i - LB_i x_i)] ,$$

s.t

$$\left.\begin{array}{l} \alpha^t - \beta^t + \gamma^t M = e_{i_k}^t \\ \alpha \geq 0, \beta \geq 0, \gamma \; random \; sign \end{array}\right\} \quad \textbf{(A.1)}$$

and

$$\left.\begin{array}{l} \alpha^t - \beta^t + \gamma^t M = -e_{i_k}^t \\ \alpha \geq 0, \beta \geq 0, \gamma \; random \; sign \end{array}\right\} \quad \textbf{(A.2)}$$

Where e_{i_k} denotes the $\boldsymbol{i_k}$ column of the identity matrix of order $\boldsymbol{n}$, and $\alpha, \beta \; and \; \gamma$ are the dual vectors associated with constraints (B.2), (B.3), and (B.4).

References

[1] http://neon.vb.cbs.nl/casc/tau.htm
[2] Matteo Fischetti & José Juan Salazar González (2000) Models and Algorithms
[3] Giessing, S. and Repsilber, D. (2002). Tools and Strategies to Protect Multiple Tables with the GHQUAR Cell Suppression Engine. In: Inference Control in Statistical Databases, Springer (Lecture Notes in Computer Science), 2316, 181-192.s
[4] Thomas Lengauer. Combinatorial Algorithms for Integrated Circuit Layout,chapter 8, pages 427{446. Applicable Theory in Computer Science. John Wiley And Sons, 1990.
[5] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers, Boston, 1997
for Optimizing Cell Suppression in Tabular Data with Linear Constraints, Journal of the American Statistical Association, 95:451, 916-928.
[6] Nicolas, J. (2010). La gestion du secret dans les tableaux diffusant des statistiques d'entreprises. In: La Lettre du SSE, 65 (French working paper).
[7] http://www.mcs.anl.gov/events/workshops/iasds11/papers/PID1994457.pdf
[8] http://blog.cloudera.com/blog/2011/09/snappy-and-hadoop/
[9] Jo~ao Pedro Pedroso DCC-FC & LIACC, Universidade do Porto R. Do Campo Alegre 823, 4150-180 Porto, Portugal