Logic and Optimization Techniques for an Error Free Data Collecting

Renato Bruni[‡] and Antonio Sassano[‡]

March 20, 2001

Abstract

In this paper an approach is proposed to the problem of automatic detection and correction of inconsistent or out of range data in a general process of data collecting. Such errors are usually detected by formulating a set of rules which must be respected in order the data records to be declared correct. A first relevant point of the proposed procedure is that the set of rules itself is checked for inconsistency or redundancy, by encoding it into a propositional logic formula, and solving a sequence of Satisfiability problem. This set of rules is then used to detect erroneous data. In the subsequent phase of error correction, the above set of rules must be satisfied, but the erroneous data should be altered as little as possible. Moreover, the statistical distribution of correct data should be preserved. As a second relevant point, such problem is modeled by encoding the rules with linear inequalities, and solving a sequence of set covering problems. The proposed procedure is tested by performing the entire process of error detection and correction in the case of a Census.
Keywords: Automatic error detection and correction, Satisfiability, Set covering, Statistic projections.

1 Introduction

When dealing with a large number of collected information, a relevant problem arises: perform all the requested elaboration considering only correct data. Examples of data collecting are for instance cases of statistical investigations, marketing analysis, experimental measures, etc. Corresponding examples of data elaboration could therefore be calculating statistical parameters, tracing consumers profiles, estimating unknown measured parameters, etc. Data correctness is a crucial aspect of data quality, and, in practical cases, it has always been a very computationally demanding problem.

Seldom data are single values. Generally, they are structured into sets of values, whose elements have a specific meaning, and are binded by specific relations. This set of p values v_i (the data) for a set of p fields f_i (their meaning) is usually called a

[‡]Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Buonarroti 12 - 00185 Roma, Italy. E-mail: {bruni,sassano}@dis.uniroma1.it

record R (among other, in the field of databases). We will therefore consider records in the following form.

$$R = \{f_1 = v_1, f_2 = v_2, \ldots, f_p = v_p\}$$

The problem of *error detection* is generally approached by formulating a set of rules that the records must respect in order to be reliable, or *consistent*. In the absence of further information, consistent records are declared *correct*. Instead, inconsistent records are declared *erroneous*. The more accurate and careful the rules are, the more truthful individuation of correct and erroneous data can be achieved. Such rules can involve the value of a single field (e.g. a value v_i must be within a set V) or a combination of values within a record (e.g. a value v_i must be equal to a value v_j plus a value v_k).

A first problem arising from this fact is the validation of such set of rules. In fact, the rules could contain some contradiction among themselves, or some rule could be implied by some other. This could result in erroneous records to be declared correct, and vice versa. This point is discussed in more detail is in section 2. The above problem of checking the set of rules against inconsistencies and redundancies is transformed in a Propositional Logic problem. This is done by encoding the rules in clauses, as explained in section 3. A sequence of propositional Satisfiability problems (SAT for short) is therefore solved, as illustrated in section 4. This procedure allows, moreover, to check if a new rule is in contradiction with the previous ones, or if they already imply it. This will reveal its importance in a phase of updating.

By choosing this clausal representation, the detection of erroneous records simply becomes the problem of testing if a propositional logic formula is satisfied by a truth assignment for its logical variables. See section 5 for details.

Since generally information collecting has a cost, we would like to somehow utilize erroneous records as well, by performing an *error correction* [20]. During such phase, erroneous records are changed in order to satisfy the above rules. This should be done by keeping as much as possible the correct information contained in such erroneous records. Two general principles should be followed: to apply the minimum changes to erroneous data, and to modify as less as possible the marginal and joint frequency distribution of the data [11]. This is described in more detail in section 6. The above problem is modeled by encoding the rules by linear inequalities, and solving a sequence of set covering problems, as explained in section 7.

The proposed procedure is tested by performing the entire process of error detection and correction in the case of a real world Census. The application and part of the data were kindly provided by the Italian National Statistic Institute (ISTAT). Results are in section 8.

2 Data Collecting through Questionnaires

Our attention will be focused on the problem of statistic projections carried out by processing answers obtained through a collection of questionnaires. We stress that such problem is used just as an example to apply our methodology, and of course does

not exhaust field of application of the proposed procedure. A record, in this case, is the set of the answers to one questionnaire Q.

$$Q = \{f_1 = v_1, f_2 = v_2 \ldots, f_p = v_p\}$$

We will consider, in particular, the case of a census of population. Examples of fields f_i are `age` or `marital status`, corresponding examples of values v_i are `18` or `single`. Fields can be distinguished in quantitative and qualitative ones. Roughly speaking, a quantitative field require its value to be either a real number a in some interval $[a_1, a_2]$, or an integer number n in some set N, or at least a value belonging to an ordered set. In a quantitative field, in fact, the order operators '$<$', '$\leq$', '$>$', '$\geq$' must be defined. On the other hand, a qualitative field require its value d to be a member of some discrete set $D = \{d_1, d_2, \ldots, d_n\}$.

Errors, or, more precisely, inconsistencies between answers or out of range answers, can be due to the original compilation of the questionnaire, or can be introduced during any later phase of information processing, such as data input or conversion. Inconsistent questionnaires could contain information that deeply modifies the aspects of interest (just think of maximum or minimum of some value), and thus, without their detection, our statistical investigation would produce erroneous results. We can distinguish between stochastic errors and systematic errors. Stochastic errors are randomly introduced, and can therefore be unpredictable and have in general low or no correlation. On the other hand, systematic errors consists in a repetition of the same error. This can be, for instance, due to some defect in the questionnaire. Of course, both of these kinds of errors must be identified.

Generally, real world statistics involve such a high number of questionnaires that an automatic procedure to carry out error detection is needed. Usually, National Statistic Offices perform the task of detecting inconsistencies by formulating rules that must be respected by every correct set of answers. More precisely, rules are written in form of *edits*. An edit expresses the error condition. The following example will clarify this point.

Example 2.1. An inconsistent answer can be to declare

`marital status` as `married` and `age` as `10` years old.

The rule to detect this kind of errors could be the following

if `marital status` is `married`, `age` must be not less than, say, `14`.

The rule must be put in form of an edit, which expresses the error condition. Since we have the error if `marital status` = `married` and `age` $<$ `14`, the edit for this example would therefore be

$$(\texttt{marital status} = \texttt{married}) \wedge (\texttt{age} < \texttt{14})$$

The set of all the edits is sometimes called the set of edit rules, or Check plan, or Compatibility plan, of a statistical investigation. Such set of edits is used to split

the set of all questionnaires in the two subsets of correct ones and erroneous ones. Questionnaires which verify the condition defined in at least one edit are declared erroneous. We will say that such questionnaires activate the edits.

Obviously, the individuation of the set of edits itself plays a crucial role. In fact, the set of edits must be free from inconsistency (i.e. edits must not be in contradiction each other), and, preferably, from redundancy (i.e. do not contain edit which are logically implied from other edits). In the case of real questionnaires, edits can became very numerous. The cardinality of the set of edits, in fact, increases with the number of questions in the questionnaire. Moreover, a high number of edits allows a more accurate error detection. Test for contradictions and redundancies must be automatically performed as well. Therefore, a form of edits representation that can be treated by automatic elaboration is needed.

Many commercial software systems deal with the problem of questionnaires correction, and they make use of a variety of different (and sometimes *naive*) edits encoding and solution algorithm [1, 25, 22]. In practical case, however, they suffer from severe limitations, due to the inherent computational complexity of the problem. Some methods ignore edit testing, and just separate erroneous questionnaires from correct ones. Their limitations are that results are incorrect if edits are incorrect, and edits updating turns out to be very difficult. This cause that number of edits must be small enough to be validated by inspection by a human operator.

On the other hand, other methods try to check for contradiction and redundancy by generating all implied edits, such as the 'Fellegi Holt' procedure [11]. Their limitation is that, as the number of edits slightly increases, they produce very poor performance. This happens, of course, because of the huge demand of computational resources required for generating all implied edits, whose number exponentially grows with the number of original edits. The above limitations prevented to now the use of a set of edits whose cardinality is above a certain value. Another serious drawback is that simultaneous processing of quantitative and qualitative fields is seldom allowed.

3 A Logical Representation of the Set of Edits

The usefulness of logic or Boolean techniques is proved by many approaches to similar problems of information representation ([5, 23] among others). This should not be surprising, when considering the role of the science of Logic. A representation of the set of edit by means of first-order logic is not new. This methodology turns out to be equivalent to the 'Fellegi-Holt' one [4], with consequent computational limitations. In this paper we propose an edit encoding by means of the simpler propositional logic. Treatment of numerical data is performed by a process called binarization [5].

A propositional logic formula is composed of propositions, i.e. logical variables (also called binary, or Boolean, variables), which assume values in the set $\{True, False\}$, or, equivalently, $\{1, 0\}$, and of the logical connectives $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$, with their usual meaning of 'and', 'or', 'implies', 'is equivalent'. Propositions can be positive (a logical variable α) or negative (a negated logical variable $\neg\alpha$). Every Propositional Logic formula can be put in conjunctive normal form (CNF), namely a conjunction ($\wedge$) of disjunctions ($\vee$). A CNF formula $\mathcal{F}$, with n logical variables and m clauses, has the

following general structure:

$$(\alpha_{i_1} \vee ... \vee \alpha_{j_1} \vee \neg\alpha_{k_1} \vee ... \vee \neg\alpha_{n_1}) \wedge \ldots \wedge (\alpha_{i_m} \vee ... \vee \alpha_{j_m} \vee \neg\alpha_{k_m} \vee ... \vee \neg\alpha_{n_m})$$

Given truth values ($True$ or $False$) to the logical variables, we have a truth value for the whole formula. A formula $\mathcal{F}$ is satisfiable if and only if there exists a truth assignment that makes the formula $True$ [9, 10, 16, 24]. If this does not exist, the formula $\mathcal{F}$ is unsatisfiable.

The problem of testing satisfiability of propositional formulae in conjunctive normal form, named SAT, plays a protagonist role in mathematical logic and computing theory. Actually, it is fundamental in Artificial Intelligence, Expert Systems, Deductive Database theory, due to its ability of formalizing deductive reasoning, and thus solving logic problems by means of automatic computation. It is known to be NP-complete [13]. Satisfiability problems indeed are used for encoding and solving a wide variety of problems arisen from different fields, e.g. VLSI logic circuit design and testing, programming language project, computer aided design. A SAT formulation can be used to solve the problem of logical implication, i.e. to detect if a given proposition is logically implied by a set of propositions [17, 14, 19, 8].

In the case of questionnaires, every edit can be encoded in a propositional logic clause. Moreover, since the edit have a very precise syntax, this encoding can be done by an automatic procedure. The set of edits E, written by the person entrusted of this task at the ISTAT, and according to the grammar used by them, is therefore transformed in a CNF propositional formula $\mathcal{E}$, following the sequence of steps listed below and described in further subsections:

Edit propositional encoding procedure

1. *Identification of the domains D_f for each one of the p field f, considering that we are dealing with errors.*
2. *Identification of subsets $S_f^1, S_f^2, \ldots$ in every domain D_f, defined by breakpoints, or cut points, $b_f^1, b_f^2, \ldots$ obtained from the edits.*
3. *Identification of equivalent subsets $S_f^j, S_k^q, \ldots$, and definition of equivalence classes $C_f^h = [S_f^j]$.*
4. *Definition of n logical variables $\alpha_{[S_f^j]}$ to represent the equivalence classes $[S_f^j]$.*
5. *Expression of all the edits by means of clauses defined over the introduced logical variables $\alpha_{[S_f^j]}$.*
6. *Identification of congruency clauses to supply the information not contained in edits.*

Note that the above procedure is merely formal, i.e. not depending by the meaning of the involved propositions. Therefore, it can be entirely performed by means of automatic symbolic elaboration, without the need for an interpretation phase.

3.1 Identification of the domains for the fields

In this step, the totality of possible answers need to be identified, including incorrect ones and blank answer. Such possibilities depend, of course, on the nature of the field (qualitative or quantitative), but also on the typographical aspect of the question in the questionnaire (for instance, single choice question, text box question, etc.). Such sets of possible answer, for the generic field f, will be indicated as D_f. D_f can include intervals of real numbers, sets of elements, etc. A generic value belonging to domain D_f will be indicated as $v_f \in D_f$.

Example 3.1. Consider a `field marital` status represented as follows:

Marital status:
☐ single ☐ married ☐ separate ☐ divorced ☐ widow

Answer can vary only on a discrete set of possibilities in mutual exclusion, or, due to errors, be missing or not meaningful (for instance when we have more than one choice). Both latter cases are expressed with the value `blank`.

$$D_{\texttt{marital status}} = \{\texttt{single}, \texttt{married}, \texttt{separate}, \texttt{divorced}, \texttt{widow}, \texttt{blank}\}$$

Example 3.2. Consider a field `age` represented as follows:

Age: _____ years

Due to errors, the totality of possible answers can be any number or be `blank`. Although this may seem too pessimistic, note that similar choices improve robustness of the procedure. We have

$$D_{\texttt{age}} = (-\infty, +\infty) \cup \{\texttt{blank}\}$$

From the above example we can see that a quantitative field can have a domain whose elements are not only numbers. To perform such identification of the domains D_f, a characterization of the qualitative or quantitative nature of every field and of its typographical aspect must be given in input to the procedure.

3.2 Identification of the breakpoints for the fields

Edits are propositions involving one or more values $v_{f_1} \in D_{f_1}, v_{f_2} \in D_{f_2}, \ldots$ for one or more fields $f_1, f_2, \ldots$. As told, they state the error condition. In our application, they may have one of the two following logical structure:

$$f_1 < relation > v_{f_1}$$

$$(f_1 < relation > v_{f_1}) < logic\ connective > (f_2 < relation > v_{f_2})$$

where $< relation >$ is one of '=', '<', '>', '$\leq$', '$\geq$', '$\in$', etc., and $< logic\ connective >$ is one of $\Rightarrow$, $\Leftrightarrow$, $\wedge$ (see Example 2.1.). Of course, order relations are used only in the case of ordered domains (quantitative fields).

Values v_f appearing in the edits are called *breakpoints*, or *cut points*, for the domains. They represent the logical *watershed* between values of the domain. Such particular values will be indicated with b_f^j. All the breakpoints b_f^j can be automatically detected by reading the edits.

We can observe that the expression $(f < relation > v_f)$ represents a set of values of the domain D_f. This can be a single values $d_f \in D_f$ (when the relation is '='), or a sets of values with cardinality > 1, or an interval of numbers. In all the above cases, anyway, $(f < relation > v_f)$ denotes a subset of domain D_f, and, in order to avoid too many case distinctions, they will all be called subsets $S_f^j \subseteq D_f$. We congruently have

$$D_f = \bigcup_j S_f^j$$

In order to identify all subsets S_f^j, the breakpoints b_f^j are used to partition domain D_f according to the edits. Some domains can be also very fragmented.

Example 3.3. For the field `marital status`, by reading an imaginary set of edits, we have the following breakpoints

$$\begin{aligned} b^1_{\texttt{marital status}} &= \texttt{single} \\ b^2_{\texttt{marital status}} &= \texttt{married} \\ b^3_{\texttt{marital status}} &= \texttt{separate} \\ b^4_{\texttt{marital status}} &= \texttt{divorced} \\ b^5_{\texttt{marital status}} &= \texttt{widow} \\ b^6_{\texttt{marital status}} &= \texttt{blank} \end{aligned}$$

and, by using the breakpoints and the edits to cut the domain $D_{\texttt{marital status}}$, we have the subsets

$$\begin{aligned} S^1_{\texttt{marital status}} &= \{\texttt{single}\} \\ S^2_{\texttt{marital status}} &= \{\texttt{married}\} \\ S^3_{\texttt{marital status}} &= \{\texttt{separate}\} \\ S^4_{\texttt{marital status}} &= \{\texttt{divorced}\} \\ S^5_{\texttt{marital status}} &= \{\texttt{widow}\} \\ S^6_{\texttt{marital status}} &= \{\texttt{blank}\} \end{aligned}$$

Example 3.4. For the field `age`, by reading an imaginary set of edits, we have the following breakpoints

$$b^1_{\texttt{age}} = \texttt{0}$$
$$b^2_{\texttt{age}} = \texttt{14}$$
$$b^3_{\texttt{age}} = \texttt{18}$$
$$b^4_{\texttt{age}} = \texttt{26}$$
$$b^5_{\texttt{age}} = \texttt{120}$$
$$b^6_{\texttt{age}} = \texttt{blank}$$

and, by using the breakpoints and the edits to cut the domain $D_{\texttt{age}}$, we have the subsets

$$S^1_{\texttt{age}} = (-\infty, \texttt{0})$$
$$S^2_{\texttt{age}} = [\texttt{0}, \texttt{14})$$
$$S^3_{\texttt{age}} = [\texttt{14}, \texttt{18})$$
$$S^4_{\texttt{age}} = \{\texttt{18}\}$$
$$S^5_{\texttt{age}} = (\texttt{18}, \texttt{26})$$
$$S^6_{\texttt{age}} = [\texttt{26}, \texttt{120}]$$
$$S^7_{\texttt{age}} = (\texttt{120}, +\infty)$$
$$S^8_{\texttt{age}} = \{\texttt{blank}\}$$

Note that, for real valued numerical fields, depending on the relations in the edit ('<', '>' '≤', '≥'), subsets are intervals close, open, left close, left open, etc.

3.3 Individuation of equivalent classes of subsets

Depending on the edits, some subsets S^a_f, S^b_f, ... can be equivalent. This happens when, given a value d_f for a field f, the following alternative cases

$$d_f \in S^a_f,\ d_f \in S^b_f,\ \ldots$$

activate exactly the same edits for every other combination of values for other fields. This means that, if d_f is in $S^a_f \cup S^b_f \cup \ldots$, changing its value still in $S^a_f \cup S^b_f \cup \ldots$ can never change correctness result for any questionnaire. By considering such equivalence relationship, we introduce equivalence classes for that. The above equivalence class will be indicated as $[S^a_f]$. Such equivalent subsets, can be identified from the edits as follows. Given a group of subsets $d_f \in S^a_f$, $d_f \in S^b_f$, ..., if all the edits where they appear are identical except for $< relation >$ and value d_i (the two elements which define the subset of the field f - see edit structure in subsect. 3.2) $d_f \in S^a_f$, $d_f \in S^b_f$, ... are equivalent. Equivalence classes can therefore be automatically identified.

Example 3.5. If all the edits where `married` and `separate` appear would be

`marital status` = `married` $\wedge$ `age` $<$ `14`
meaning: if `marital status` is `married`, `age` must be not less than `14`.

`marital status` = `separate` $\wedge$ `age` $<$ `14`
meaning: if `marital status` is `separate`, `age` must be not less than `14` (must be married to be separate).

the two subsets `married` and `separate` would be equivalent. Since, instead, they also appear in other edits which do not satisfy the above condition, `married` and `separate` are not equivalent. In this case the equivalence condition never holds for the field `marital status`, hence we have no equivalent subset in it. Therefore, there is a different class for every subset.

Example 3.6. On the contrary, for the field `age`, some subsets are equivalent. In particular, edits representing the concept *out of normal* are

(`age` $<$ `0`)
meaning: `age` cannot be less than `0`.

(`age` $>$ `120`)
meaning: `age` cannot be more than `120`.

(`age` = `blank`)
meaning: `age` cannot be `blank`.

The above subsets does not appear in any other edit, and the above edits are identical except for $< relation >$ and value d_i, hence the subsets $\{(-\infty, 0)\}, \{(120, +\infty)\}, \{\texttt{blank}\}$ are equivalent and collapse in to the same class. Moreover, also the subsets $\{18\}$ and $\{(18, 26)\}$ results to be equivalent. There are no further equivalent subsets. Altogether, for the field `age`, we have the classes

$$C^1_{\texttt{age}} = [\{\texttt{blank}\}] = \{(-\infty, 0), (120, +\infty), \texttt{blank}\}$$
$$C^2_{\texttt{age}} = [[0, 14)] = \{[0, 14)\}$$
$$C^3_{\texttt{age}} = [[14, 18)] = \{[14, 18)\}$$
$$C^4_{\texttt{age}} = [(18, 26)] = \{18, (18, 26)\}$$
$$C^5_{\texttt{age}} = [[26, 120]] = \{[26, 120]\}$$

3.4 Definition of logic variables

The logic variables are used to encode the information about which equivalence class contains the value of every field. This can be done in several ways, and pursuing different aims[1] We choose the following, with the aim to produce an easier CNF formula. The definition of logical variables is slightly different in the case of qualitative or quantitative fields.

For every qualitative field with n classes, we use $n-1$ logic variables α_i corresponding to $n-1$ classes. If the qualitative field f has a value belonging to the class C^j_f we put $\alpha_{C^j_f} = True$. The same holds for the other classes of the field, except for one, for instance the last one C^n_f. When the field f has a value in this last class C^n_f, we put all variables at $False$.

Example 3.7. The field `marital status` is divided in 6 equivalence classes (see example 3.5). We therefore have 6-1 = 5 logical variables $\alpha_{[\texttt{single}]}$, $\alpha_{[\texttt{married}]}$, $\alpha_{[\texttt{separate}]}$, $\alpha_{[\texttt{divorced}]}$, $\alpha_{[\texttt{widow}]}$, as shown below.

[1] h equivalence classes could for instance be encoded by $\lceil \log_2 h \rceil$ logic variables.

$$
\begin{array}{lcl}
v_{\texttt{marital status}} \in [\texttt{single}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = True \\ \alpha_{[\texttt{married}]} = \alpha_{[\texttt{separate}]} = \\ = \alpha_{[\texttt{divorced}]} = \alpha_{[\texttt{widow}]} = False \end{cases} \\
v_{\texttt{marital status}} \in [\texttt{married}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = False \\ \alpha_{[\texttt{married}]} = True \\ \alpha_{[\texttt{separate}]} = \alpha_{[\texttt{divorced}]} = \alpha_{[\texttt{widow}]} = False \end{cases} \\
v_{\texttt{marital status}} \in [\texttt{separate}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = \alpha_{[\texttt{married}]} = False \\ \alpha_{[\texttt{separate}]} = True \\ \alpha_{[\texttt{divorced}]} = \alpha_{[\texttt{widow}]} = False \end{cases} \\
v_{\texttt{marital status}} \in [\texttt{divorced}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = \alpha_{[\texttt{married}]} = \alpha_{[\texttt{separate}]} = False \\ \alpha_{[\texttt{divorced}]} = True \\ \alpha_{[\texttt{widow}]} = False \end{cases} \\
v_{\texttt{marital status}} \in [\texttt{widow}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = \alpha_{[\texttt{married}]} = \\ = \alpha_{[\texttt{separate}]} = \alpha_{[\texttt{divorced}]} = False \\ \alpha_{[\texttt{widow}]} = True \end{cases} \\
v_{\texttt{marital status}} \in [\texttt{blank}] & \Rightarrow & \begin{cases} \alpha_{[\texttt{single}]} = \alpha_{[\texttt{married}]} = \alpha_{[\texttt{separate}]} = \\ = \alpha_{[\texttt{divorced}]} = \alpha_{[\texttt{widow}]} = False \end{cases}
\end{array}
$$

For every quantitative field with n classes, we use $n-1$ variables. The difference with former case is that these variables do not correspond to classes. They correspond instead to nested intervals, all of them having as initial point the smaller feasible value for that field, and as final point the identified breakpoints. Externally from the bigger nested interval it remains the 'out of range' class. This choice for variable association results, in our case, in shorter clauses. In fact, in most of edits appear similar intervals. They can therefore be expressed with one variable instead of more ones.

If the quantitative field f has a value in one of the nested intervals `[a,b]` we put $\alpha_{[\texttt{a,b}]} = True$. The other variable must be set accordingly, as illustrated below. The same holds for the other classes of the field, except for one, for instance the last one C_f^n. When the field has a value in the 'out of range' class, we put all variables at $False$.

Example 3.8. The field `age` is divided in 5 equivalence classes (see example 3.6). We therefore have 5-1 = 4 logical variables $\alpha_{[0,14)}$, $\alpha_{[0,18)}$, $\alpha_{[0,26)}$, $\alpha_{[0,120]}$, as illustrated below.

$$
\begin{array}{lcl}
v_{\texttt{age}} \in [[0, 14)] & \Rightarrow & \left\{ \begin{array}{l} \alpha_{[0,14)} = \alpha_{[0,18)} = \\ = \alpha_{[0,26)} = \alpha_{[0,120]} = True \end{array} \right. \\
v_{\texttt{age}} \in [[14, 18)] & \Rightarrow & \left\{ \begin{array}{l} \alpha_{[0,14)} = False \\ \alpha_{[0,18)} = \alpha_{[0,26)} = \alpha_{[0,120]} = True \end{array} \right. \\
v_{\texttt{age}} \in [[18, 26)] & \Rightarrow & \left\{ \begin{array}{l} \alpha_{[0,14)} = \alpha_{[0,18)} = False \\ \alpha_{[0,26)} = \alpha_{[0,120]} = True \end{array} \right. \\
v_{\texttt{age}} \in [[26, 120]] & \Rightarrow & \left\{ \begin{array}{l} \alpha_{[0,14)} = \alpha_{[0,18)} = \alpha_{[0,26]} = False \\ \alpha_{[0,120]} = True \end{array} \right. \\
v_{\texttt{age}} \in [\texttt{blank}] & \Rightarrow & \left\{ \begin{array}{l} \alpha_{[0,14)} = \alpha_{[0,18)} = \\ = \alpha_{[0,26]} = \alpha_{[0,120]} = False \end{array} \right.
\end{array}
$$

We therefore have the following graphical representation of the nested intervals.

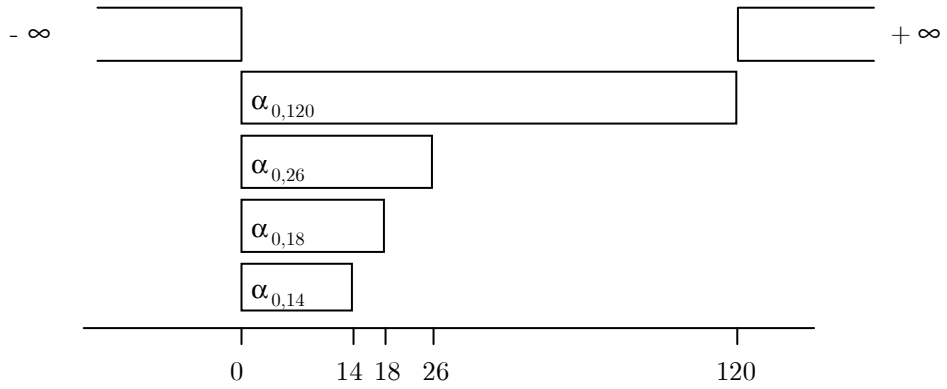

Figure 1: Logic variables used to encode the field `age`.

3.5 Encoding of edits as clauses

So far, every edit can be encoded in clauses by using the defined variables. Every expressions ($f_i < relation > v_{f_i}$) can in fact be substituted by the corresponding logical variable, obtaining therefore a sequence of logic variables connected by logic operators, hence a logic formula.

We are interested in producing clauses which are satisfied by consistent answers. Being edits the description of the error condition, we now need to negate the obtained logic formula. This negated logic formula can be then transformed in CNF format, by following simple syntactical logic rules. The so generated CNF formula is satisfied by every set of correct questionnaires answers, and are not satisfied by every set of inconsistent or out of range answer. With the particular edit structure considered,

every edit produces one and only one clause. However, in a general case, this could not hold, but the procedure works fine as well.

Example 3.9. Following the above example 3.1, the given edit is

$$\texttt{marital status} = \texttt{married} \wedge \texttt{age} < 14$$

By substituting the logical variables, we have the following logic formula

$$\alpha_{[\texttt{married}]} \wedge \alpha_{[0,14)}$$

By negating it, and applying De Morgan's law, we obtain the following clause

$$\neg\alpha_{[\texttt{married}]} \vee \neg\alpha_{[0,14)}$$

3.6 Identification of congruency clauses

In addition to information given by edits, there is information which is not contained in edits, and that a human operator would consider obvious, but which must be provided to an automatic elaboration. In our case, we need to express that fields must have one and only one value, and therefore other clauses, named congruency clauses, need to be added.

By using $n-1$ variables for n equivalence classes, there is no need to add clauses expressing that we must have a value being in at least one class, because it does not exist a truth assignment for variables that doesn't verify this. Instead, it must be expressed that the value for a field must be in only one class. Considering the case of qualitative fields (such as the example of `marital status`), we have the n variables corresponding to the $n+1$ disjoint classes. The above condition becomes that only one variable can be true. This is imposed by adding clauses constituted by all the possible couples of negated variables. Their number is therefore $\binom{n}{2}$ (number of combination of class 2 of n objects).

Example 3.10. In the case of the qualitative field `marital status`, we have 5 variables, hence the congruency clauses are $\binom{5}{2} = 10$, as follows:

$$\begin{array}{ccc}
\neg\alpha_{[single]} & \vee & \neg\alpha_{[married]} \\
\neg\alpha_{[single]} & \vee & \neg\alpha_{[separate]} \\
\neg\alpha_{[single]} & \vee & \neg\alpha_{[divorced]} \\
\neg\alpha_{[single]} & \vee & \neg\alpha_{[widow]} \\
\neg\alpha_{[married]} & \vee & \neg\alpha_{[separate]} \\
\neg\alpha_{[married]} & \vee & \neg\alpha_{[divorced]} \\
\neg\alpha_{[married]} & \vee & \neg\alpha_{[widow]} \\
\neg\alpha_{[separate]} & \vee & \neg\alpha_{[divorced]} \\
\neg\alpha_{[separate]} & \vee & \neg\alpha_{[widow]} \\
\neg\alpha_{[divorced]} & \vee & \neg\alpha_{[widow]}
\end{array}$$

Considering now the case of quantitative fields (such as the example of `age`), we have n variables corresponding to n nested subsets. As observed above, we can have either

the case of a quantitative field divided into nested intervals of reals, or the case of a quantitative field divided into nested sets of integer numbers. We will discuss the procedure speaking about nested intervals, but, of course, the case of nested sets of integers is completely analogous.

The congruency condition becomes that, if a variable $\alpha_{[a_1,b_1]}$, corresponding to an interval $[a_1, b_1]$, is $True$, also all the variables $\alpha_{[a_2,b_2]}, \alpha_{[a_3,b_3]}, \ldots$, corresponding to all the intervals $[a_2, b_2], [a_3, b_3], \ldots$, containing $[a_1, b_1]$, must be $True$. The above condition is imposed by adding $(n-1) + (n-2) + \ldots + 1 = \binom{n}{2}$ clauses expressing

$$\alpha_{[a_1,b_1]} \Rightarrow \alpha_{[a_2,b_2]}, \alpha_{[a_1,b_1]} \Rightarrow \alpha_{[a_3,b_3]}, \ldots, \alpha_{[a_2,b_2]} \Rightarrow \alpha_{[a_3,b_3]}, \ldots,$$

Converting in CNF by applying elementary logic rules ($\alpha_a \Rightarrow \alpha_b$ is equivalent to $\neg\alpha_a \vee \alpha_b$), we obtain

$$\neg\alpha_{[a_1,b_1]} \vee \alpha_{[a_2,b_2]}, \neg\alpha_{[a_1,b_1]} \vee \alpha_{[a_3,b_3]}, \ldots, \neg\alpha_{[a_2,b_2]} \vee \alpha_{[a_3,b_3]}, \ldots,$$

Example 3.11. In the case of the quantitative field `age`, we have 4 variables corresponding to 4 nested subsets, hence the congruency clauses are $\binom{4}{2} = 6$, as follows:

$$\begin{array}{lcl}
\neg\alpha_{[0,14)} & \vee & \alpha_{[0,18)} \\
\neg\alpha_{[0,14)} & \vee & \alpha_{[0,26)} \\
\neg\alpha_{[0,14)} & \vee & \alpha_{[0,120]} \\
\neg\alpha_{[0,18)} & \vee & \alpha_{[0,26)} \\
\neg\alpha_{[0,18)} & \vee & \alpha_{[0,120]} \\
\neg\alpha_{[0,26)} & \vee & \alpha_{[0,120]}
\end{array}$$

So far, given the set of edits, we have a set of m clauses, and, given the set of answers to a questionnaire, we have a truth assignment for the n logical variables. By construction, a record which does not activate any edit, will satisfy all the clauses, and a record which activates some edits will not satisfy the corresponding clauses. We therefore have that the truth assignment given by a record must satisfy all the clauses to be declared correct. We will hence consider the conjunction of all the clauses, that is a CNF formula $\mathcal{E}$, and say, briefly, that the questionnaire Q must satisfy $\mathcal{E}$ to be declared correct.

4 Edits Validation

As told, edits must be free from inconsistency (i.e. edits must not be in contradiction each other), and, preferably, from redundancy (i.e. do not contain edit which are logically implied from other edits). The test for contradictions and redundancies is very hard for a human operator, and impossible above a certain number of edits. On the other hand, when such test is automatically performed, it historically turns out to be very computationally demanding.

A diffuse approach follows the so-called 'Fellegi-Holt' methodology [11]. This consists in checking for contradiction and redundancy by generating all the (new) edits

which are implied by the given edits. The generation of all implied edits allows to check if the given edits imply some *hidden* rule that should not hold. Unfortunately, but predictably, the number of implied edits is exponential in the number of given edits. This is the main reason for which this methodology, although theoretically irreproachable, is recently recognized to be not applicable for many real-world problems.

Edit representation by means of first-order logic was already proposed, for instance in [4]. This allows a formal and convenient description of the edit and the operations. However, something analogous to the generation of all implied edits must be performed in this case also. The computational complexity of the problem remains exponential.

Being our proposal to perform edit validation in the case of real world problems, we must get rid of the generation of all implied edits. By means of a propositional logic representation, the problem of checking for inconsistency and redundancy can be formalized as follows.

4.1 Complete Inconsistency in the Set of Edits

When every possible set of answers to the questionnaire is declared incorrect, we have the situation called *complete inconsistency* of the set of edits. In fact, reminding that edits describe the inconsistent situation, let us consider the following example.

Example 4.1. The following is a situation of Complete Inconsistency. This is, of course, a very simple and evident one. More complex ones, involving dozens of edits, are not so easily visible. Below every edit its meaning is explained.

`seaside house` = `no`
meaning: everybody must have a seaside house.

`mountain house` = `no`
meaning: everybody must have a mountain house.

(`seaside house` = `yes`) $\wedge$ (`mountain house` = `yes`)
meaning: it is not allowed to have both seaside and mountain house.

Using these edits, every questionnaire fails the check, because it cannot exist any set of answers which appear consistent. Edits are always activated.

In a large set of edits, or in a phase of edits updating performed by people different from the original edit writers, the situation of complete inconsistency may occur.

Claim 4.1. *By encoding edits in clauses, complete inconsistency correspond to a CNF formula that cannot be satisfied, namely an unsatisfiable formula.*

Complete inconsistency can therefore be detected by checking the satisfiability of the whole CNF formula.

Moreover, when having a complete inconsistency and in order to restore consistency, it would be very useful to know which are the inconsistent edits. This corresponds to selecting which part of the unsatisfiable CNF cause the unsolvability, i.e. a minimal unsatisfiable subformula (MUS) [7, 12, 18]. This could be done by means of the used SAT solver, which, in the case of unsatisfiable instances, is able to select a subset of clauses which are still unsatisfiable, and thus causes the unsatisfiability of the whole instance [7]. Therefore, the inconsistent edits are the ones corresponding to such clauses, and should be changed (by the human adviser who writes the edits, of course). The usefulness of the individuation of such inconsistent subset of edits is easily understandable when thinking of the prospect of checking, for instance, a dozen of edits instead of one thousand.

4.2 Partial Inconsistency in the Set of Edits

More insidious because less easy to detect without an automatic procedure in a large set of edits is the situation of *partial inconsistency* of the set of edits. This happens when some questionnaires, which are correct, are declared erroneous, only due to a particular value $v_f^\dagger$ of a single field f. In the intentions of the edit writer, $v_f^\dagger$ was a feasible value for the field f. Due to an unfortunate edit combination, however, there is an hidden and unwanted implied edit which forbids the value $v_f^\dagger$ for the field f.

$v_f^\dagger$ will be called a *sentence value*. When a correct questionnaire contains a sentence value, it is (erroneously) declared incorrect, due to edit's fault. For questionnaires not containing sentence values, partial inconsistency does not cause any problem. Let us consider indeed the following example.

Example 4.2. The following is a situation of partial inconsistency. Below every edit its meaning is explained.

(`annual income` $\geq$ `1000`) $\Rightarrow$ (`seaside house` = `no`)
meaning: if annual income is greater then or equal to 1000, then the subject must have a seaside house.

(`annual income` $\geq$ `2000`) $\Rightarrow$ (`mountain house` = `no`)
meaning: if annual income is greater then or equal to 2000, then the subject must have a mountain house.

(`seaside house` = `yes`) $\wedge$ (`mountain house` = `yes`)
meaning: it is not allowed to have both seaside and mountain house.

Using the above edits, every questionnaires where the subject has an `annual income` $\geq$ `2000` is declared erroneous, even if it should not. In fact, that answer necessarily activates the edits. Note that, for `annual income` $<$ `2000`, this partial inconsistence does not show any effect, and error detection proceeds in the correct way. The value `2000`, (and any other greater value) is a sentence value for the field `annual income`.

We can have more than one sentence value $v_f^\dagger$ forbidden by the same unwanted implied edit. The set of all values forbidden by an unwanted implied edit is in fact constituted

by the subset S_f^j containing $v_f^\dagger$, together with all other subsets belonging to the same equivalence class (see sec. 3). The logical variable corresponding to the equivalence class $[S_f^j]$ will be called *sentence variable* $\alpha_f^\dagger$ for the field f. The set of all the values forbidden by one unwanted implied edit will be called *sentence set* $S_f^\dagger$.

$$S_f^\dagger \quad = \quad \bigcup_{k:\ S_f^k \in [S_f^j],\ v_f^\dagger \in S_f^j} S_f^k$$

We will therefore say that we have partial inconsistency with regard to the set $S_f^\dagger$ for the field f, or, equivalently, with regard to the variable $\alpha_f^\dagger$ for the field f. Note that we can have more unwanted implied edits forbidding more variables of the same field.

In case of partial inconsistency, the CNF formula obtained from the edits is satisfiable. However, if for field f we have a sentence value, this corresponds to fixing the sentence variable $\alpha_f^\dagger$ to $True$.

Claim 4.2. *If we fix $\alpha_f^\dagger$ to $True$, and remove, as standard, satisfied clauses and all negated occurrence of that variable, the resulting formula becomes unsatisfiable.*

Basing on the above result, partial inconsistency with regard to any single variable α_k of the CNF formula can be detected by checking the satisfiability of all the CNF formulae obtained by independently fixing $\alpha_k = True$, for $k = 1, \ldots, n$.

Partial inconsistency with respect to a single variable α_k, will be called first-level partial inconsistency. Partial inconsistency with respect to a couple of variables α_k and α_h, will be called second-level partial inconsistency, and so on. We don't check all levels, otherwise we could not get rid from the exponential complexity that affects other procedures. What we do is to check all first-level partial inconsistencies, and provide a tool that allows to check, if desired, higher-level partial inconsistencies. Note that neither 'Fellegi-Holt' methods check such partial inconsistencies, since they do dot lead to a complete contradiction when deriving all implied edits, and therefore are not automatically signaled. With 'Fellegi-Holt' methods, in fact, they could only be found by a (hypothetic) human inspector which examines every implied edit generated.

Moreover, in order to restore consistency, the used SAT solver is still able to select a subset of clauses which are unsatisfiable, and thus causes the situation of partial inconsistency. Individuation of inconsistent edits can therefore be carried out also in the case of partial inconsistency.

4.3 Redundancy in the set of edits

Some edits could be logically implied by others, being therefore redundant. It would be obviously preferable if we could remove them, because decreasing the number of edits while maintaining the same power of inconsistency detection can simplify the whole process and make it less error prone.

Example 4.3. The following is a (very simplified) situation of edit redundancy. Below every edit its meaning is explained.

(`role` = `head of the house`) $\wedge$ (`annual income` $<$ `100`)
meaning: head of the house must have an annual income greater then or equal to 100.

`annual income` $<$ `100`
meaning: everybody must have an annual income greater then or equal to 100.

The first edit is clearly redundant.

A SAT formulation is generally used to solve the problem of logical implication, i.e., given a set of statements S called axioms, to decide whether another statement s logically derives from them, in symbols $S \Rightarrow s$. Representing statements S and s with clauses, we have $S \Rightarrow s$ if and only if $S \cup \neg s$ is an unsatisfiable formula [17, 19]. In our case this means:

Claim 4.3. *The clausal representation of an edit e_j is implied by the clausal representation of a set of edits E if and only if the CNF formula obtained by $E \cup \neg e_j$ is unsatisfiable.*

It can be consequently checked if an edit is redundant by removing its clausal representation e_j from the CNF formula, by adding its negation $\neg e_j$ to the formula, and by testing if the resulting formula is unsatisfiable. The redundancy of every edit can be checked by independently applying to each one of them the above operation

5 Individuation of Erroneous Records

Once we have a valid set of edit rules, they are used to detect erroneous records, in our case questionnaires. A correct questionnaire will be indicated with Q^r (right) and an erroneous one with Q^w (wrong). By using the propositional logic representation, this trivially becomes the problem of checking if the truth assignment corresponding to each questionnaire Q satisfies the CNF formula $\mathcal{E}$ obtained from the set of edits plus the congruency clauses. This operation can be performed with an extremely small computational effort. It results therefore suitable to check even a very large number of questionnaires.

6 The Problem of Imputation

After detection of erroneous records, if information collecting has no cost, we could just cancel erroneous records and collect new information until we have enough correct records. Since usually information collecting has a cost, we would like to use also the correct part of information contained in the erroneous records. This means changing the erroneous records in order to try to restore the unknown correct values. Such operation is called *imputation*.

We will call *original data* the unknown data that we would have if we had no errors, and *original frequency distributions* the unknown frequency distributions that we would have if we had no errors. We will correspondingly call *original questionnaire* Q^o the questionnaire that we would have if we had no errors.

In our case, therefore, given an erroneous questionnaire Q^w, the imputation process consists in changing some of his values in order to obtain a *corrected questionnaire* Q^c which satisfies the formula $\mathcal{E}$.

Imputation should be done by keeping as much as possible the correct information contained in erroneous data. Two general principles should be followed [11]:

- To apply the minimum changes to erroneous data.
- To modify as less as possible the marginal and joint original frequency distribution of the data.

The above principles frequently clash. We give an example to clarify this point.

Example 6.1. Consider this case of error: someone whose `age` is `70` and `marital status` is `married`. The original questionnaire would be

$Q^o = \{$ ... `age` = `70`, ... `marital status` = `married`, ... $\}$

However, the subject forgets to write the zero of `70` and writes `7`. The questionnaire we actually have is

$Q^w = \{$ ... `age` = `7`, ... `marital status` = `married`, ... $\}$

Such record is immediately detected as erroneous, since we have an edit which says 'if `marital status` is `married`, `age` must be $\geq$ `14`'. Suppose that, in virtue of some procedure (presented below), we know that the errors is in the field `age`. We therefore want to correct the value of `age`. The point is how to correct that. Imagine in fact that we have many records which are in the above situation, in the sense that their original values for `age` were any value above `14`, but the erroneous values we have on the questionnaire are all below `14`. If we just try to correct `age` by changing it as less as possible, we put all of such values at `14`, as follows.

$Q^c = \{$ ... `age` = `14`, ... `marital status` = `married`, ... $\}$

The statistical distribution of the original `age` would therefore be remarkably altered, by having a wrong peak in `14`. Note that, even if we try to correct `age` by choosing a random value above `14` which has the same frequency distribution of the `age` of married people (distribution obtained from the correct questionnaires) we could alterate the joint frequency distribution of the field `age` with respect to other fields. To continue the example, imagine that our subject (who is 70 years old), has, for the field `work`, the value `retired`. Imagine that the random value used to correct `age` is `50` (quite plausible). We have:

Q^c ={ ... `age` = `50`, ... `marital status` = `married`, ... `work` = `retired`}

We can observe that the original (unknown) joint frequency distribution of the two fields `age` and `work` is altered. The same could happen for the other original joint frequency distributions.

There are two main approaches to the above problem of imputation: the deterministic and the probabilistic one. Such approaches can sometimes be combined. Given an erroneous record, a deterministic imputation reconstruct the record with a deterministic procedure, renouncing to keep the original unknown joint frequency distributions. Note that errors can be stochastic or systematic. Stochastic errors are randomly introduced, and can therefore be unpredictable and have in general low or no correlation. Systematic errors consists in a repetition of the same error. This can be, for instance, due to some structural defect in the questionnaire. Deterministic methods are valid in the case of systematic errors, being in fact a systematic correction. Deterministic imputation just require a priori decisions, and therefore do not present specific computational difficulties.

A probabilistic imputation, on the contrary, for every erroneous record, tries to correct it by choosing new values which are not predetermined. The same erroneous records can therefore be corrected in more than one way. Such methods are generally preferred in the case of stochastic errors, because they assure a more uniform data correction, and can salvage the original frequency distributions. The drawback of probabilistic procedures usually is their computational burden.

6.1 Error Localization

A first problem arising when a questionnaire is declared erroneous, is to locate the error, namely to understand which are the erroneous fields. We could assume that the erroneous fields are the smallest set of fields that, if changed, permit to restore consistency, in the sense of not activating the edits anymore. This assumption is based on the fact that, when error is something unintentional, the more likely event is that errors are the smaller number. This is coherent with the principle of minimum change. In addiction to the above, one can argue that some fields can be more reliable than others. This means that the probability they are erroneous is lower. What is generally done is to give, for every field f_i, a measure $r_i \in \mathbb{R}_+$ of the reliability, which corresponds to a "preference" in keeping unchanged the value of field f_i. We define the total cost of a correction as the sum of the reliabilities r_i of the fields to modify f_i^w. By calling W the set of the f_i^w, the cost of such correction is

$$c\left(W\right) = \sum_{i:f_i^w \in W} c_i$$

Therefore, the problem of error localization is the following. Given the erroneous questionnaire Q^w and the CNF formula $\mathcal{E}$ to be satisfied, we want to find a set W of fields f_i^w such that:

- The corrected questionnaire Q^c satisfying $\mathcal{E}$ can be obtained from the erroneous one Q^w by changing (only and all) the values of the $f_i^w \in W$.
- The total cost of required changes $c(W)$ is minimum.

We can in fact have more than one set of fields whose imputation can restore consistency, and we are interested in the one which changes as few as possible Q^w.

As for the values to put in such fields, two approaches are generally considered. One is to generate the imputed values by means of some (stochastic) function, although this could change the original unknown frequency distribution of the data (see example 6.1). The second is to use a donor questionnaire Q^d.

6.2 Imputation through a Donor

A donor questionnaire Q^d is a correct questionnaire which, according to some distance function $d(Q^w, Q^d) \in \mathbb{R}_+$, is the nearest one to the erroneous questionnaire Q^w we want to correct. Therefore, it represents a subject which has very similar characteristics. Given Q^w and Q^d, we simply copy the values of the fields f_i^w that we need to change from the donor Q^d to the erroneous Q^w. This procedure is generally recognized to cause a low alteration of the original frequency distributions.

Example 6.2. Consider this case of the error in example 6.1: someone whose `age` is `70` and `marital status` is `married`. The erroneous questionnaire we have is

Q^w = { ... `age` = `7`, ... `marital status` = `married`, ... }

Assume that the field `marital status` has a cost $c_{\texttt{marital status}} = 8$, and the field `age` has a cost $c_{\texttt{age}} = 3$. This means that the answer to the field `marital status` is considered more reliable than the answer to the field `age`. We find that the set W of minimum cost is just the field {`age`}. We therefore want to perform the imputation of the field `age`.

Assume that, if we consider the values of all the fields of Q^w, the subject appears to be an elderly man. Let therefore assume that, by searching for a correct questionnaire at minimum distance (considering all fields) from Q^w we find another elderly man:

Q^d = { ... `age` = `72`, ... }

We can now proceed with the imputation of the field `age` from the donor. This restores consistency. We obtain

Q^c ={ ... `age` = `72`, ... `marital status` = `married`, ... }

Note that, in this case, the corrected questionnaire Q^c is very similar to the original one Q^o. This does not happen by chance, but is due to the mechanism of imputation trough a donor.

However, two problems arise from the use of a donor. The first is that, if we have not enough correct questionnaires to find a donor which is close to Q^w, the imputed values can be not so similar to the original ones. The second is that, by using a donor, we are not guaranteed that the set of erroneous fields of minimum cost W is enough to restore consistency of Q^w. In fact, W is that set of fields at minimum cost that can be changed to restore consistency, but the use of a donor does not let to change such fields as much as we like. We could need to take more fields from the donor, or we could need to take the fields of another set $W' \neq W$, before restoring consistency. However, we have the guarantee that, since the donor is a correct questionnaire, there exists in Q^d at least one set of fields whose values are able to restore consistency of Q^w. The second problem is related to the first, since, having a very numerous set of correct questionnaires, the case when we need to copy from the donor a set of fields different from W before restoring consistency is rare. Moreover, the number of fields we could need to add in this case is low. Consider the following (very rough) example.

Example 6.3. We have an erroneous questionnaire Q^w

Q^w = { ... `age` = `17`, ... `car` = `no`, ... `city of residence` = `aaa`, ...
..., `city of work` = `bbb`, ... `time to go to work` = `20`, ... }

where the set of erroneous fields of minimum cost is { `age`, `car` }, with a total cost of 5.5. Imagine we could restore consistency only if, for these two fields, we have `age`$\geq$ `18` and `car` = `yes`. Searching for a donor, however, we find Q^d such that

Q^w = { ... `age` = `16`, ... `car` = `yes`, ... `city of residence` = `aaa`, ...
... `city of work` = `aaa`, ... `time to go to work` = `20`, ... }

If we proceed with imputation of the two selected fields { `age`, `car` }, we do not restore consistency at all. We need to choose a different set of fields. In this case, imagine that the imputation of the set of fields { `age`, `city of work` }, with a total cost of $c = 6$, restores consistency. By taking them from the donor, we obtain:

Q^w = { ... `age` = `17`, `car` = `yes`, `city of residence` = `aaa`, ...
... `city of work` = `aaa`, ... `time to go to work` = `20`, ... }

Therefore, the problem of imputation trough a donor is the following. Given the erroneous questionnaire Q^w, the donor questionnaire Q^d, and the CNF formula $\mathcal{E}$ to be satisfied, we want to find a set D of fields f_i^d such that:

- The corrected questionnaire Q^c satisfying $\mathcal{E}$ can be obtained from the erroneous one Q^w by copying from the donor Q^d (only and all) the values of the $f_i^d \in D$.
- The total cost of the correction $c(D)$ is minimum.

In this case also we can have more than one set of fields whose imputation can restore consistency, and we are interested in the one which changes as few as possible Q^w. Due to the motivations noted above, we have that $c(W) \leq c(D)$. Variants to the above procedure are possible (the number of donors, how to choose a donor), but the spirit of the imputation trough a donor remains the same.

7 A Set Covering Formulation

We can observe that, in both the cases of the use of an imputation function or of the use of a donor, we actually want to find a set of changes of minimum cost such as we can restore consistency, i.e. satisfy the CNF formula $\mathcal{E}$. The above problem can be modeled as a weighted set covering problem [21].

Given a ground set S of n elements s_i, and a collection $\mathcal{A}$ of m subsets A_j of S (for example, the collection may consists of all subsets of size $k \leq n$), the set covering problem is the problem of taking a set of elements s_i of minimum cardinality such as we have at least one element for every A_j. A little more general problem is the following. Given a ground set S of n elements s_i, each one with a cost $c_i \in \mathbb{R}_+$, and a collection $\mathcal{A}$ of m sets A_j of elements of S, the weighted set covering problem is the problem of taking the set of elements s_i of minimum total weight such as we have at least one element for every A_j. Note that the first problem is a special case of the second when all the costs are 1.

Let a^j be the incidence vector of A_j, i.e. a vector in $\{0,1\}^n$ whose i-th component a_i^j is 1 if $s_i \in A_j$, and 0 if $s_i \notin A_j$. Consider a vector of variables $x \in \{0,1\}^n$ which is the incidence vector of the set of the elements s_i we take. We can give the following mathematical model for the above problems.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i x_i \\ \text{s. t.} \quad & \sum_{i=1}^{n} a_i^j x_i \geq 1 \qquad\qquad j = 1 \ldots m \\ & x \in \{0,1\}^n \end{aligned} \tag{1}$$

The set covering problem is a classical combinatorial optimization problem, with binary variables which assume values in {0,1}. It is of great relevance for modeling and solving a variety of problems arising form many practical fields. It is known to be NP-complete [13]. Set covering formulations are used, for instance, in the fields of telecommunications, transportation, facility location, crew scheduling, and, in general, when the problem has the structure of a set of *something* that must be *covered*.

In order to work in the field of binary optimization, we now transform the logic variables α_i taking values in $\{True, False\}$ into binary variables x_i taking values in {0,1}. The difference is only formal, and the conversion is straightforward. While the operations defined on the logical variables where $\{\wedge, \vee, \ldots\}$, the operations defined on the binary variables are $\{+, -, \ldots\}$ In particular, a positive literal α_i corresponds to a binary variable x_i, and a negative literal $\neg\alpha_i$ corresponds to a negated binary variable $\bar{x}_i$. Note that $\bar{x}_i$ and x_i must have opposite values, just like $\neg\alpha_i$ and α_i.

A set of answers to a questionnaire, which corresponded to a truth assignment in $\{True, False\}^n$, will now correspond to a binary vector in $\{0,1\}^n$. The following propositional clause c_j

$$(\alpha_i \vee \ldots \vee \alpha_j \vee \neg\alpha_k \vee \ldots \vee \neg\alpha_n)$$

will now correspond to the following linear inequality, by defining the set A_π of the logical variables which appear positive in c_j, and the set A_ν of the logical variables which appear negative in c_j, and the corresponding incidence vectors a^π and a^ν

$$\sum_{i=1}^{n} a_i^\pi x_i + \sum_{i=1}^{n} a_i^\nu \bar{x}_i \geq 1$$

For each clause we have an inequality of the above type. If we write all of them, we obtain that the logic formula $\mathcal{E}$ that the truth assignment must satisfy

$$(\alpha_{i_1} \vee ... \vee \alpha_{j_1} \vee \neg\alpha_{k_1} \vee ... \vee \neg\alpha_{n_1}) \wedge \ldots \wedge (\alpha_{i_m} \vee ... \vee \alpha_{j_m} \vee \neg\alpha_{k_m} \vee ... \vee \neg\alpha_{n_m})$$

corresponds now to the following system of linear inequalities that the binary vector must satisfy.

$$\begin{pmatrix} a^\pi_{1,1} & ... & a^\pi_{1,n} \\ & ... & \\ a^\pi_{m,1} & ... & a^\pi_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} a^\nu_{1,1} & ... & a^\nu_{1,n} \\ & ... & \\ a^\nu_{m,1} & ... & a^\nu_{m,n} \end{pmatrix} \begin{pmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_n \end{pmatrix} \geq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

Example 7.1. Suppose that the truth assignment corresponding to the (correct) questionnaire Q is

$$\{\alpha_1 = False, \alpha_2 = False, \alpha_3 = True\}$$

and that the logic formula $\mathcal{E}$ used to detect erroneous questionnaires is

$$(\neg\alpha_1 \vee \alpha_2 \vee \neg\alpha_3) \wedge (\neg\alpha_1 \vee \neg\alpha_2) \wedge (\alpha_2 \vee \alpha_3)$$

The binary vector corresponding to the questionnaire Q is

$$\{x_1 = 0, x_2 = 0, x_3 = 1\}$$

and the system of linear inequalities corresponding to $\mathcal{E}$ is

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{pmatrix} \geq \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Moreover, in order to take into account the reliability of every field, we estimate a vector of n costs $\{c_1, \ldots, c_n\}$ corresponding to the variables $\{\alpha_1, \ldots, \alpha_n\}$. We pay c_i when we change α_i. Note that, since every field corresponds to more variables, such costs are not directly the reliability of a field. On the other hand, an estimation of an individual cost for every variable allows more subtile evaluations.

We can now model the imputation problems as follows.

7.1 Error Localization

In the case of error localization, we have

- The binary vector $e = \{e_1, \ldots, e_n\} \in \{0,1\}^n$ corresponding to the erroneous questionnaire Q^w.
- The binary variables $x = \{x_1, \ldots, x_n\} \in \{0,1\}^n$ and their complements $\bar{x} = \{\bar{x}_1, \ldots, \bar{x}_n\} \in \{0,1\}^n$, with the coupling constraints $x_i + \bar{x}_i = 1$. The variables x represent the truth assignment corresponding to the corrected questionnaire Q^c that we want to find.
- The system of linear inequalities $A^\pi x + A^\nu \bar{x} \geq 1$, with $A^\pi, A^\nu \in \{0,1\}^{m \times n}$, that e does not satisfy. We know that such system has binary solutions, since $\mathcal{E}$ is satisfiable and has more than one solution.
- The vector $c' = \{c_1, \ldots, c_n\} \in \mathbb{R}^n_+$ of costs that we pay for changing e. We pay c_i for changing e_i.

We introduce furthermore a vector of binary variables $y = \{y_1, \ldots, y_n\} \in \{0,1\}^n$ representing the changes we introduce in e. We have

$$y_i = \begin{cases} 1 & \text{if we change } e_i \\ 0 & \text{if we keep } e_i \end{cases}$$

According to the principle of the minimum change, we want to change the erroneous questionnaire Q^w minimizing the total cost of the changes. This can be expressed as

$$\min_{y_i \in \{0,1\}} \sum_{i=1}^{n} c_i y_i = \min_{y \in \{0,1\}^n} c'y \tag{2}$$

We want a corrected questionnaire Q^c which satisfies the system of inequalities

$$A^\pi x + A^\nu \bar{x} \geq 1 \tag{3}$$

A key issue is that there is a relation between variables y and x (and consequently $\bar{x}$). This depends on the values of e, as follows:

$$y_i = \begin{cases} x_i \quad (= 1 - \bar{x}_i) & \text{if } e_i = 0 \\ 1 - x_i \quad (= \bar{x}_i) & \text{if } e_i = 1 \end{cases} \tag{4}$$

In fact, when $e_i = 0$, to keep it unchanged means to put $x_i = 0$. Since we do not change, $y_i = 0$. On the contrary, to change it means to put $x_i = 1$. Since we change, $y_i = 1$. Altogether, $y_i = x_i$.

When, instead, $e_i = 1$, to keep it unchanged means to put $x_i = 1$. Since we do not change, $y_i = 0$. On the contrary, to change it means to put $x_i = 0$. Since we change, $y_i = 1$. Altogether, $y_i = 1 - x_i$.

By using the above result, we can express the problem of error localization with the following formulation. Our objective function (2) becomes

$$\min_{x_i,\bar{x}_i \in \{0,1\}} \quad \sum_{i=1}^{n}(1-e_i)c_i x_i + \sum_{i=1}^{n} e_i c_i \bar{x}_i \tag{5}$$

Subject to the following constraints

$$\begin{aligned} &A^{\pi}x + A^{\nu}\bar{x} \geq 1 \\ &x_i + \bar{x}_i = 1 \\ &x, \bar{x} \in \{0,1\}^n \end{aligned}$$

We will call *satisfiability* constraints the first kind of constraints, deriving in fact from the propositional satisfiability problem. We will call *coupling* constraints the second kind of constraint. The third kind are the *binary* constraints over the variables. The above formulation is a set covering problem, as defined in (1).

7.2 Imputation through a Donor

The case of imputation through a donor is very similar. We have

- The binary vector $e = \{e_1, \ldots, e_n\} \in \{0,1\}^n$ corresponding to the erroneous questionnaire Q^w.
- The binary vector $d = \{d_1, \ldots, d_n\} \in \{0,1\}^n$ corresponding to the donor questionnaire Q^d.
- The binary variables $x = \{x_1, \ldots, x_n\} \in \{0,1\}^n$ and their complements $\bar{x} = \{\bar{x}_1, \ldots, \bar{x}_n\} \in \{0,1\}^n$, with the coupling constraints $x_i + \bar{x}_i = 0$. x corresponds to the corrected questionnaire Q^c that we want to find.
- The system of linear inequalities $A^{\pi}x + A^{\nu}\bar{x} \geq 1$, with $A^{\pi}, A^{\nu} \in \{0,1\}^{m \times n}$, that e does not satisfy. We know that such system has binary solutions, since $\mathcal{E}$ is satisfiable and has more than one solution.
- The vector $c' = \{c_1, \ldots, c_n\} \in \mathbb{R}_+^n$ of costs that we pay for changing e. We pay c_i for changing e_i.

We use, as before, a vector of binary variables $y = \{y_1, \ldots, y_n\} \in \{0,1\}^n$ representing the elements that we copy from d to e. We have

$$y_i = \begin{cases} 1 & \text{if we copy } \ d_i \text{ in } e_i \\ 0 & \text{if we keep } \ e_i \end{cases}$$

According to the principle of the minimum change, we want to change the erroneous questionnaire Q^w minimizing the total cost of the changes. This can be expressed as

$$\min_{y_i \in \{0,1\}} \quad \sum_{i=1}^{n} c_i y_i \quad = \quad \min_{y \in \{0,1\}^n} c'y \tag{6}$$

We want a corrected questionnaire Q^c which satisfies the system of inequalities

$$A^\pi x + A^\nu \bar{x} \geq 1 \tag{7}$$

In this case also, there is a relation between variables y and x (and consequently $\bar{x}$). This depends on the values of e and d, as follows:

$$y_i = \begin{cases} x_i \quad (= 1 - \bar{x}_i) & \text{if } e_i = 0 \text{ and } d_i = 1 \\ 1 - x_i \quad (= \bar{x}_i) & \text{if } e_i = 1 \text{ and } d_i = 0 \\ 0 & \text{if } e_i = d_i \end{cases} \tag{8}$$

In fact, when $e_i = 0$ and $d_i = 1$, not to copy the element means to put $x_i = 0$. Since we do not change, $y_i = 0$. On the contrary, to copy the element means to put $x_i = 1$. Since we change, $y_i = 1$. Altogether, $y_i = x_i$.

When, instead, $e_i = 1$ and $d_i = 0$, not to copy the element means to put $x_i = 1$. Since we do not change, $y_i = 0$. On the contrary, to copy the element means to put $x_i = 0$. Since we change, $y_i = 1$. Altogether, $y_i = 1 - x_i$.

Finally, when $e_i = d_i$, we have no gain from copying the element, and therefore we do not change. Altogether, $y_i = 0$.

By using the above result, we can express the problem of imputation through a donor with the following formulation. Our objective function (2) becomes

$$\min_{x_i, \bar{x}_i \in \{0,1\}} \quad \sum_{i=1}^{n} (1 - e_i) d_i c_i x_i + \sum_{i=1}^{n} e_i (1 - d_i) c_i \bar{x}_i \tag{9}$$

Subject to the following constraints

$$\begin{aligned} & A^\pi x + A^\nu \bar{x} \geq 1 \\ & x_i + \bar{x}_i = 1 \\ & x, \bar{x} \in \{0,1\}^n \end{aligned}$$

Similarly with the former case, we will call *satisfiability* constraints the first kind of constraints, deriving in fact from the propositional satisfiability problem. We will call *coupling* constraints the second kind of constraint. The third kind are the *binary* constraints over the variables. Also the above formulation is a set covering problem, as defined in (1).

The above problems are therefore solved with the volume algorithm [2, 3], which is a recently proposed and very effective combinatorial optimization algorithm.

8 Implementation

We first implemented a series of procedures in order perform the automatic conversion of the set of edits into a CNF formula. After that, in virtue of modeling our problems as classical optimization problems, we could take advantage of the huge amount of research done in the fields of satisfiability and set covering solvers.

The satisfiability problems are solved by means of an enumeration procedure called Adaptive Core Search (ACS)[7]. Such solver, in fact, turns out to be a very effective

one, as proved in [6]. Moreover, in the case of unsatisfiable instances, ACS is able to select a subset of clauses which are still unsatisfiable, and thus cause the unsatisfiability. This is a key feature in our case, since, in the cases of inconsistencies, we need to correct them, and this is possible only when the set of 'culprit' edits can be located.

All the edits validation process was implemented to run sequentially, and interactively asking, for every inconsistence of redundancy found, whether stop the process and let the edit-writer mend the set of edit, or ignore it and continue.

The set covering problems are then solved by implementing the Volume Algorithm [2, 3]. This very effective procedure recently developed by Barahona is an extension of the subgradient algorithm, with the key feature is that is able to produce primal as well dual solutions. This gives a fast method for producing approximated solutions for large scale linear programs.

We added a simple heuristic in order to obtain an integer solution to our set covering problem, together with a bound allowing to verify the quality of such solution. Such heuristic consists in a rounding of the fractionary solution. Rounding is deterministic (and just cuts at 1/2) for some fractionary values, while being probabilistic for particular combinations of fractionary values. The choice of the Volume Algorithm made possible to solve problems whose size is not solvable by a commercial Branch-and-Bound solver (Xpress).

9 Results

We performed the process of edits validation and data imputation in the case of a Census of Population. The set of edits are kindly provided by the Italian National Statistic Institute (ISTAT). In the course of this work, we actually did several tests, by considering different sets of edits. They ranged from 100 to 400 edits. Such procedure were considered to be quite representative of how a real census proceeds.

All the described procedure were implemented in C++ and tested on a Pentium II 450MHz PC under MS Windows Operating System.

In this census, the data of every family were collected by using a single questionnaire. Individuals were identified by number, and edits were originally written with this structure. Edits have been rewritten identifying individuals by roles, which are: `head of the family`, `consort`, `father of head`, `mother of head`, `brother/sister of head`, `son`, `additional son`, `father in law`, `mother in law`, `daughter in law`, `son in law`, `grandchild`. This was done by taking obviously care of maintaining the exact meaning of the original edits. This choice produces a more compact set of edits, while its congruency and redundancy properties remain the same.

Being a questionnaire for an entire family, we can have similar edits repeated for every member of the family. Edit replication, called explosion, is the first automatically performed step.

It is then applied the procedure of automatic conversion of edits into clauses. The implemented software takes in input a file containing the set of edits, a file containing a description of the fields, a file containing lists of all feasible values of qualitative fields, and a file containing the list of family roles. The procedure identifies logical

variables, according to outlined procedure, and gives in output a file listing the used logical variables, with their meaning, and the CNF formula which encodes the set of edits. The generation of the logic formula is not a costing operation.

For the problem we deal with, resulting formulas range from 315 variables and 650 clauses to 450 variables and 1100 clauses. Since this kind of problems are solved quite easily, in order to test the limits of the procedure, we moreover considered artificially generated satisfiability instances of bigger size. They ranged from 1000 variables and 5000 clauses to 15000 variables and 75000 clauses.

9.1 Edit Validation

Our algorithm begins with solving the satisfiability problem for the CNF formula, in order to detect complete inconsistency for the set of edits. After this, in order to detect all 1-level partial inconsistency, the procedure goes ahead fixing in turn every variable to $True$ and then all the variables of every field to $False$, and solving at every step the resulting SAT instance.

The test for redundancies is then made by negating in turn every clauses, and solving at every step the resulting SAT instance.

Hence, for every instance with n variables and m clauses, the implemented algorithm solves in cascade about $1 + n + n/10 + m$ satisfiability problems of non trivial size. Although such problems do not reveal to be structurally hard, the computational burden is substantial. Therefore, there is the need of an efficient SAT solver. We used ACS, a recently proposed and very efficient SAT solver, which uses an enumeration scheme with a new adaptive technique [7]. In the following tables we report number of variables (n) and number of clauses (m) of the propositional CNF formula, the number of problem we had to solve (# of problems), and total time for solving all of them (time).

In table 1 we report results on formulas which are the encoding of the real census edits. In table 2 we report results on artificially generated formulas.

n	m	# of problems	time
315	650	975	0.99
350	710	1090	1.35
380	806	1219	1.72
402	884	1321	2.21
425	960	1428	2.52
450	1103	1599	3.24

Table 1: Results of the edit validation procedure on real sets of edits .

Our solver is able, in the case of an unsatisfiable instance, of providing a set of clauses which cause the unsolvability, in order to understand were is the inconsistency. In the analyzed set of real edits, which were supposed to be error free, a partial inconsistency was found, that was due to one of the edit concerning divorced people, as explained below. Married people must be at least `14` years old, and a divorce procedure takes at

least four years. It results that divorced people must be at least 18 years old. There was an edit representing this, and it was erroneously written

$$(\texttt{marital status} \in \{\texttt{single}, \texttt{married}, \texttt{separate}, \texttt{widow}\}) \wedge (\texttt{age} < 18)$$

The correct edit should have been the following.

$$(\texttt{marital status} = \texttt{divorced}) \wedge (\texttt{age} < 18)$$

Such problem could of course cause errors during the phase of individuation of erroneous records.

By checking for redundancy, several clauses resulted redundant, hence the corresponding edits are already implied by the rest of the edits.

Further tests, performed after deliberately introduction of inconsistency or redundancy in the set of edits, lead in the totality of the cases to their detection.

n	m	# of problems	time
1000	5000	6101	15
3000	15000	18301	415
5000	25000	30501	1908
8000	40000	48801	7843
10000	50000	61001	16889
15000	75000	91501	>36000

Table 2: Results of the edit validation procedure on artificially generated sets of edits .

In the case of artificially generated instances, partial inconsistencies and redundancies were found. Their description is not significative, being artificially generated problems. They were used in order to understand which size of problems the proposed procedure is able to treat.

Afterwards, detection of erroneous questionnaires answers have been performed, as a trivial task. It proceeds converting the set of answers in values for the logical variables, and simply testing if such truth assignment satisfies the CNF formula obtained from the edits.

9.2 Data Imputation

The procedure for data imputation was tested on simulated questionnaires, and by using the two sets of real edits and the artificially generated ones. We considered both the problems of error localization and imputation through a donor. For each set of edits we considered various simulated erroneous answers. In particular, we considered the percentage of activated edits for the erroneous answer. Since errors are usually a small part of the answers, we realistically considered small edit activation percentages (1%, 2%, 3%, etc.). Note that a set of edit corresponding to n logic variables and m clauses corresponds here to a set covering formulation with $2n$ variables and m satisfiability constraints plus n coupling constraints, altogether $m + n$.

We solved the set covering formulation by means of a set covering solver based on the Volume Algorithm. Moreover, since this is an approximate procedure, we compared its results with the commercial Branch-and-Bound solver Xpress. In some cases of high error percentage, the solver based on the Volume Algorithm could not find a feasible integer solution. In those cases we reported "-". We report a different table for each instance, with time and solution value in all the above cases. Problems of error localization follow.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	0.01	3.18	43.0	43.0
0.9%	0.10	3.18	221.6	221.6
1.3%	0.10	3.45	328.4	328.4
2.4%	0.14	2.86	729.5	729.5
5.0%	0.22	3.50	300.8	300.8

Table 3: Error localization procedure on a real set of edits. The set covering instance has 400 var. and 1400 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	0.04	1.91	58.6	58.6
0.7%	0.10	1.91	108.6	108.6
1.0%	0.11	2.54	140.1	140.1
1.6%	0.16	1.90	506.7	506.7
2.5%	0.20	2.50	1490.1	1490.1
3.5%	0.23	2.98	2330.8	2330.8

Table 4: Error localization procedure on a real set of edits. The set covering instance has 480 var. and 1880 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.7%	0.51	2.44	463.2	463.2
1.5%	0.68	3.02	394.5	394.5
2.1%	0.55	3.32	612.5	612.5
3.6%	0.98	3.73	1254.8	1254.8
6.7%	1.00	4.40	2341.2	2341.2

Table 5: Error localization procedure on a real set of edits. The set covering instance has 800 var. and 3200 const.

From the above tables, we can observe that real problems are solved efficiently both by VA and B&B. Surprisingly, VA reaches in all cases the optimal integer solution, given by the B&B. Times are very small in both cases, and they increase with error percentage (and, of course, with the size of the problem).

In addition, we tested the procedure on artificially generated instances, corresponding to the artificially generated satisfiability instance considered in the case of edit validation.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	0.66	87.90	329.4	329.4
1.0%	1.30	81.88	1010.4	1010.4
1.1%	0.73	97.71	952.9	952.9
1.8%	1.92	109.05	1982.0	1982.0
4.3%	3.65	87.16	5549.8	5549.8

Table 6: Error localization procedure on an artificially generated set of edits. The set covering instance has 2000 var. and 6000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	11.60	3813.4	1273.7	1273.6
0.8%	4.75	3477.7	2266.1	2266.1
1.0%	5.10	2796.9	3604.3	3604.3
2.1%	4.76	4117.2	6064.2	6064.2
4.0%	11.63	3595.0	1544.7	1544.6

Table 7: Error localization procedure on an artificially generated set of edits. The set covering instance has 6000 var. and 18000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	8.27	18837.8	2018.1	2018.1
0.7%	10.87	19210.5	3817.1	3817.1
1.3%	11.80	19493.6	6409.2	6409.2
1.7%	28.58	18506.6	9673.7	9673.7
3.0%	41.35	17000.3	21742.5	21742.4

Table 8: Error localization procedure on an artificially generated set of edits. The set covering instance has 10000 var. and 30000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	9.16	>21600	3058.4	-
0.9%	20.36	>21600	6897.7	-
1.3%	40.64	>21600	11682.9	-
2.0%	66.60	>21600	20113.0	-
4.0%	73.54	>21600	37069.1	-

Table 9: Error localization procedure on an artificially generated set of edits. The set covering instance has 16000 var. and 48000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	44.44	>21600	5135.4	-
0.8%	30.61	>21600	8640.5	-
1.3%	31.42	>21600	14466.7	-
2.1%	83.41	>21600	23093.7	-
3.6%	74.64	>21600	39448.2	-

Table 10: Error localization procedure on an artificially generated set of edits. The set covering instance has 20000 var. and 60000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	35.74	>21600	6754.7	-
0.9%	47.33	>21600	12751.3	-
1.3%	107.97	>21600	20135.4	-
2.0%	94.42	>21600	31063.6	-
4.0%	186.92	>21600	66847.4	-

Table 11: Error localization procedure on an artificially generated set of edits. The set covering instance has 30000 var. and 90000 const.

From the above tables, we can observe that artificially generated problems of very big size are all solved by VA, while B&B cannot solve the bigger instance within the time limit of 6 hours. However, we begin to notice that the solution found by VA is not optimal, although the difference is numerically negligible.

Further occasional tests with higher error percentage shows that VA does not increase running time, but the heuristic is not able to find a feasible integer solution, hence we have no solution, while B&B would reach such solution but in an a prohibitive amount of time.

In the case of a donor, we considered a simulated donor, which is a correct solution very similar to the erroneous one. Results both for real and artificially generated sets of edits follow.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	0.04	0.01	43.4	43.4
0.9%	0.04	0.01	227.5	227.5
1.3%	0.04	0.01	348.7	348.7
2.4%	0.04	0.04	726.6	726.6
5.0%	0.03	0.03	365.8	365.8

Table 12: Imputation through a donor on a real set of edits. The set covering instance has 400 var. and 1400 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	0.04	0.02	63.6	63.6
0.7%	0.06	0.02	144.7	144.7
1.0%	0.06	0.02	264.5	264.5
1.6%	0.06	0.02	643.5	643.1
2.5%	0.07	0.02	1774.3	1774.1
3.5%	0.06	0.03	2369.9	2369.5

Table 13: Imputation through a donor on a real set of edits. The set covering instance has 480 var. and 1880 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.7%	0.08	0.04	280.1	280.1
1.5%	0.08	0.06	453.5	453.1
2.1%	0.08	0.06	655.4	655.0
3.6%	0.08	0.07	1378.0	1378.0
6.7%	0.10	0.07	2455.1	2455.0

Table 14: Imputation through a donor on a real set of edits. The set covering instance has 800 var. and 3200 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	0.26	0.01	331.2	331.2
1.0%	0.10	0.31	1015.2	1015.1
1.1%	0.04	0.01	952.9	952.9
1.8%	0.08	0.31	1997.1	1997.1
4.3%	-	0.20	-	5592.2

Table 15: Imputation through a donor on an artificially generated set of edits. The set covering instance has 2000 var. and 6000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	0.17	0.30	1291.6	1291.4
0.8%	0.17	0.10	2305.4	2305.4
1.0%	0.70	0.30	3660.1	3660.1
2.1%	0.38	0.30	6124.0	6123.1
4.0%	0.65	0.30	1585.7	1585.0

Table 16: Imputation through a donor on an artificially generated set of edits. The set covering instance has 6000 var. and 18000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	1.01	0.6	2120.1	2118.9
0.7%	1.30	0.6	3877.7	3877.2
1.3%	1.40	0.3	6411.7	6411.7
1.7%	1.60	0.6	9690.1	9690.1
3.0%	1.05	0.3	21823.1	21802.4

Table 17: Imputation through a donor on an artificially generated set of edits. The set covering instance has 10000 var. and 30000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	0.44	0.6	3064.2	3064.2
0.9%	4.59	0.6	6899.7	6899.6
1.3%	0.86	0.8	11702.9	11698.3
2.0%	-	0.6	-	19814.5
4.0%	-	0.9	-	36496.7

Table 18: Imputation through a donor on an artificially generated set of edits. The set covering instance has 16000 var. and 48000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.5%	2.85	0.9	5139.1	5139.0
0.8%	3.28	0.9	8688.0	8687.0
1.3%	3.41	0.9	14500.4	14478.5
2.1%	3.82	0.9	23411.7	23072.0
3.6%	-	0.9	-	44026.8

Table 19: Imputation through a donor on an artificially generated set of edits. The set covering instance has 20000 var. and 60000 const.

	Time		Value	
error	VA	B&B	VA	B&B
0.4%	3.58	1.27	6788.1	6788.1
0.9%	3.22	1.27	12760.0	12759.5
1.3%	0.97	0.9	20140.1	20140.0
2.0%	2.89	1.27	31258.1	31082.2
4.0%	-	1.59	-	67764.94

Table 20: Imputation through a donor on an artificially generated set of edits. The set covering instance has 30000 var. and 90000 const.

From the above tables, we can observe that all problems are solved is very short times. B&B is always able to find a solution within seconds. VA, on the other hand, is not able to find a solution when error increases too much. Moreover, the solution found by VA is not optimal in several cases, although the difference is very small.

This can be explained by noting that, in the problem of imputation through a donor, some variables are fixed to the value thy have in the donor. This results in a

problem with much less variables but with a high error percentage. In this conditions the preferable solution method is a complete one, such like B&B.

To summarize the results, VA outperform B&B in the case of error localization in very big instances with a moderate error. On the contrary, in the case of smaller problems with higher error (not realistic), or in some cases of imputation through a donor, B&B is more reliable.

10 Conclusions

A binary encoding is the more direct and effective representation both for records and for the set of edit rules in a process of data collecting. It allows to automatically detect inconsistencies and redundancies in the set of edit rules. Erroneous records detection is carried out with an inexpensive procedure. The proposed encoding allows, moreover, to automatically perform error localization and data imputation. The related computational problems are overcome by using state-of-the-art solvers. Approached real problems have been solved in extremely short times. Artificially generated problems are effectively solved until sizes which are orders-of-magnitude larger than the above real-world problems. Hence, noteworthily qualitative improvements in a general process of data collecting are made possible. The implemented software is tested in the case of a real Population Census. Edits are kindly provided by the Italian National Statistic Institute (ISTAT). Results are extremely encouraging.

Acknowledgments. The authors would like to thank Sonia Benzi, Francisco Barahona, Fabiana Birarelli, Carlo Mannino, Alessandra Reale.

References

[1] M. Bankier. Experience with the New Imputation Methodology used in the 1996 Canadian Census with Extensions for future Census. *UN/ECE Work Session on Statistical Data Editing*, Working Paper n.24, Rome, Italy, 2-4 June 1999.

[2] F. Barahona and R. Anbil. The Volume Algorithm: producing primal solutions with a subgradient method. *IBM Research Report* RC21103, 1998.

[3] F. Barahona and F. Chudak. Near-optimal solutions to large scale facility location problems. *IBM Research Report* RC21606, 1999.

[4] G. Barcaroli. Un approccio logico formale al problema del controllo e della correzione dei dati statistici. *Quaderni di Ricerca ISTAT* n.9/1993.

[5] E. Boros, P.L. Hammer, T. Ibaraki and A. Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[6] R. Bruni and A. Sassano. CLAS: a Complete Learning Algorithm for Satisfiability. *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"* , Technical Report 6-99, 1999.

[7] R. Bruni and A. Sassano. Finding Minimal Unsatisfiable Subformulae in Satisfiability Instances. *in proc. of 6th Internat. Conf. on Principles and Practice of Constraint Programming*, Lecture notes in Computer Science 1894, Springer-Verlag, 500–505, 2000.

[8] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Comm. Assoc. for Comput. Mach.*, 5:394–397, 1962.

[9] M. Davis and H. Putnam. A computing procedure for quantification theory. *Jour. Assoc. for Comput. Mach.*, 7:201–215, 1960.

[10] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:415–436, 1996.

[11] P. Fellegi and D. Holt. A Systematic Approach to Automatic edit and Imputation. *Journal of the American Statistical Association*, 17:35–71(353), 1976.

[12] H. Fleischner and S. Szeider. Polynomial-time Recognition of Mimimal Unsatisfiable Formulas with Fixed Clause-Variable Difference. *submitted to Elsevier Preprint*. 1999.

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[14] M.R. Genesereth and N.J. Nillson. Logical Foundation of Artificial Intelligence. *Morgan Kaufmann*, 1987.

[15] F. Harche, J.N. Hooker, and G.L. Thompson. A computational study of Satisfiability Algorithms for Propositional Logic. *ORSA Journal on Computing*, 6:423–435, 1994.

[16] R.E. Jeroslow and J. Wang. Solving Propositional Satisfiability Problems. *Annals of Mathematics and AI*, 1:167–187, 1990.

[17] R. Kowalski. *Logic for Problem solving*. North Holland, 1978.

[18] O. Kullmann. An application of matroid theory to the SAT problem. *ECCC TR00-018*, February 2000.

[19] D.W. Loveland. *Automated Theorem Proving: a Logical Basis*. North Holland, 1978.

[20] M. Masselli, M. Signore, F. Panizon. *Il sistema di controllo della qualitá dei dati*. Manuale di tecniche di indagine ISTAT, vol. 6, 1992.

[21] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. J. Wiley, New York, 1988.

[22] C. Poirier. A Functional Evaluation of Edit and Imputation Tools. *UN/ECE Work Session on Statistical Data Editing*, Working Paper n.12, Rome, Italy, 2-4 June 1999.

[23] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[24] A. Van Gelder and Y.K. Tsuji. Satisfiability testing with more reasoning and less guessing. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:559–586, 1996.

[25] W.E. Winkler. State of Statistical Data Editing and current Research Problems. *UN/ECE Work Session on Statistical Data Editing*, Working Paper n.29, Rome, Italy, 2-4 June 1999.