Errors Detection and Correction in Large Scale Data Collecting

Renato Bruni and Antonio Sassano

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

Abstract. The paper is concerned with the problem of automatic detection and correction of inconsistent or out of range data in a general process of statistical data collecting. Under such circumstances, errors are usually detected by formulating a set of rules which the data records must respect in order to be declared correct. As a first relevant point, 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 records should be altered as little as possible, and frequency distributions of correct data should be preserved. As a second relevant point, error correction is modeled by encoding the rules with linear inequalities, and solving a sequence of set covering problems. The proposed procedure is tested on a real-world case of Census.

1 Introduction

When dealing with a large amount of collected information, a relevant problem arises: perform the requested elaboration considering only correct data. Examples of data collecting are cases of statistical investigations, marketing analysis, experimental measures, etc. Our attention will be focused on the problem of statistic projections carried out by processing answers to questionnaires. We will consider, in particular, the case of a census of population. We stress that such problem is just an example to apply our methodology, but does not exhaust its field of application. A data record is a set of values v_i for a set of fields f_i. In our case, a record is the set of the answers to one questionnaire Q.

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

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. A quantitative field is a field on whose values are applied (at least some) mathematical operators (e.g. $>$, $+$), hence such operators should be defined. Examples of quantitative field are numbers (real or integer), or even the elements of an ordered set. A qualitative field requires its value to be member of a discrete set with finite number of elements. Errors, or, more precisely, inconsistencies between answers or out of range answers, can be due to the original compilation of the questionnaire, or 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. The problem of *error detection* is generally approached by formulating a set of rules that the records must respect in order to be consistent, or *correct*. Instead, inconsistent records are declared *erroneous*. Rules are generally written in form of *edits*. An edit expresses the error condition, as a conjunction of expressions $(f_i < relation > v_{f_i})$.

Example 1.1. 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: if `marital status` is `married`, `age` must be not less than, say, `14`. Hence, there is an error if `marital status` = `married` and `age` $<$ `14`, and the edit, that is the error condition, is

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

Questionnaires which verify the condition defined in at least one edit are declared erroneous. Obviously, the set of edits must be free from *inconsistency* (i.e. edits must not contradict each other), and, preferably, from *redundancy* (i.e. edits must not be logically implied by other edits). In the case of real questionnaires, edits can be very numerous, since a high number of edits allows a better quality error detection. Many commercial software systems deal with the problem of questionnaires correction. They make use of a variety of different edits encoding and solution algorithm (e.g. [1],[12],[13]). 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 divide erroneous questionnaires from correct ones. In such cases, since results are incorrect if edits contains contradictions, the number of edits must be small enough to be validated by inspection by a human operator. Moreover, edits updating turns out to be very difficult. Other methods try to check for contradiction and redundancy by generating all implied edits, such as the 'Fellegi Holt' procedure [6]. Their limitation is that, as the number of edits slightly increases, they produce very poor performance. This happens 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.

By encoding the rules in clauses, the above problem of checking the set of rules against inconsistencies and redundancies is here transformed into a *propositional logic* problem (Sect. 2). A sequence of *propositional Satisfiability* problems is therefore solved (Sect. 3). Since generally information collecting has a cost, we would like to utilize erroneous records as well, by performing an *error correction*. 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 the erroneous records (Sect. 4). The above problem is modeled by encoding the rules in linear inequalities, and solving a sequence of *set covering* problems (Sect. 5). The proposed procedure is tested by performing the 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). Additional low-level details can be found in [5].

2 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 (e.g. [3]). A representation of the set of edit by means of first-order logic is not new, with consequent computational limitations. In this paper we propose an edit encoding by means of the easier-to-solve propositional logic. A propositional logic formula $\mathcal{F}$ in *conjunctive normal form* (CNF) is a conjunction of clauses C_j, each clause being a disjunction of literals l_i, each literal being either a positive (α_i) or a negative ($\neg\alpha_i$) logic variable. By denoting the possible presence of $\neg$ by $[\neg]$, this is

$$\bigwedge_{j=1..m} \Big(\bigvee_{i=1..|C_j|} [\neg]\alpha_i \Big) \tag{1}$$

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 $True$ (i.e. a *model*). If this does not exist, $\mathcal{F}$ is *unsatisfiable*. The problem of testing satisfiability of propositional formulae in conjunctive normal form, named SAT, is well-known to be NP-complete [7], and plays a protagonist role in mathematical logic and computing theory. A SAT formulation can be used to solve the problem of logical implication, that is to detect if a given proposition is logically implied by a set of propositions [8],[9],[10]. In the case of questionnaires, every edit can be encoded in a propositional logic clause. Moreover, since edits have a very precise syntax, encoding could be performed by means of the following automatic procedure.

Edit propositional encoding procedure

1. *Identification of the domains D_f for each one of the p fields f, considering that we are dealing with errors.*
2. *Identification of k_f subsets $S_f^1, S_f^2, \ldots, S_f^{k_f}$ in every domain D_f, by using breakpoints, or cut points, b_f^j obtained from the edits, and by merging (possible) equivalent subsets within each domain D_f.*
3. *Definition of n_f logical variables $\alpha_f^1, \alpha_f^2, \ldots, \alpha_f^{n_f}$ to encode the k_f subsets S_f^j of each field f.*
4. *Expression of each edit by means of clauses defined over the introduced logical variables α_f^j.*
5. *Identification of congruency clauses to supply the information not contained in edits.*

Example 2.1. For the qualitative field `marital status`, answer can vary on a discrete set of possibilities in mutual exclusion, or, due to errors, be missing or not meaningful. Both latter cases are expressed with the value `blank`.

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

For the quantitative field `age`, due to errors, the domain is

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

Values v appearing in the edits are called *breakpoints*, or *cut points*, for the domains. They represent the logical *watershed* between values of the domain,

and will be indicated with b_f^j. Such breakpoints are used to split every domain D_f into subsets S_f^j representing values of the domain which are *equivalent* from the edits' point of view. We congruently have $D_f = \bigcup_j S_f^j$.

Example 2.2. For the field `age` 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 $D_{\texttt{age}}$, we have the subsets

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

Subsets $(-\infty, \texttt{0}), (\texttt{120}, +\infty), \{\texttt{blank}\}$, representing *out of range* values, are equivalent (can be automatically detected) and collapse in to the same subset $S^1_{\texttt{age}}$.

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

So far, subsets can be encoded with logic variables in several ways (for instance, k_f subsets can be encoded by $\lceil \log_2 k_f \rceil$ logic variables). We choose to encode the k_f subsets of every domain with $n_f = k_f - 1$ variables, with the aim to produce an easier-to-solve CNF. When the value v of field f belongs to subset S_f^j, this means $\alpha_f^j = True$ and $\alpha_f^h = False$, for $h = 1, \ldots, n_f, h \neq j$. The same holds for the other subsets of f, except for the *out of range* subset (present for every field), which is encoded by putting all variables α_f^h at $False$, for $h = 1, \ldots, n_f$.

Example 2.3. The field `marital status` is divided in 6 subsets. We therefore have 6-1 = 5 logical variables

$$\alpha_{[\texttt{single}]}, \alpha_{[\texttt{married}]}, \alpha_{[\texttt{separate}]}, \alpha_{[\texttt{divorced}]}, \alpha_{[\texttt{widow}]}$$

Now every edit can be encoded in clauses by using the defined variables. Every expressions $(f_i < relation > v_{f_i})$ can be substituted by the corresponding logical variable, obtaining a conjunction of logic variables. Since we are interested in clauses satisfied by correct records, and being edits the error condition, we need to negate such conjunction, obtaining a disjunction, hence a clause.

Example 2.4. Consider the following edit.

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

By substituting the logical variables, we have the logic formula $\alpha_{[\texttt{married}]} \wedge \alpha_{[\texttt{0,14}]}$. By negating it, and applying De Morgan's law, we obtain the following clause

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

In addition to information given by edits, there is other information that a human operator would consider obvious, but which must be provided. With our choice for variables, we need to express that fields must have one and only one value, and therefore $\binom{n_f}{2}$ (number of combination of class 2 of n_f objects) clauses, named congruency clauses, are added. Altogether, the set of edits produces a set of m clauses with n logical variables, and the set of answers to a questionnaire produces a truth assignment for such logical variables. By construction, all and only the truth assignments given by correct questionnaires satisfy all the clauses, hence the CNF formula $\mathcal{E}$. Briefly, a questionnaire Q must satisfy $\mathcal{E}$ to be correct.

3 Edits Validation

In order to check the set of edits against inconsistency and redundancy, we can study the models of $\mathcal{E}$. When every possible set of answers to the questionnaire is declared incorrect, we have the situation called *complete inconsistency* of the set of edits. When the edit inconsistency appears only for particular values of particular fields, we have the (even more insidious) situation of *partial inconsistency* of the set of edits.

Example 3.1. A very simple complete inconsistency, with edits meaning: (a) everybody must have a seaside house, (b) everybody must have a mountain house, (c) it is not allowed to have both seaside and mountain house. More complex ones, involving dozens of edits, are not so easily visible.

$$\begin{aligned} &\texttt{seaside house} = \texttt{no} && \text{(a)}\\ &\texttt{mountain house} = \texttt{no} && \text{(b)}\\ &(\texttt{seaside house} = \texttt{yes}) \wedge (\texttt{mountain house} = \texttt{yes}) && \text{(c)} \end{aligned}$$

Example 3.2. A very simple partial inconsistency, with edits meaning: (a) the subject must have a seaside house if and only if annual income is greater then or equal to 1000, (b) the subject must have a mountain house if and only if annual income is greater then or equal to 2000, (c) it is not allowed to have both seaside and mountain house. For $\texttt{annual income} < 2000$, this partial inconsistence does not show any effect, but every questionnaires where the subject has an $\texttt{annual income} \geq 2000$ is declared erroneous, even if it should not. We have a partial inconsistency with respect to the subset $\texttt{annual income} \geq 2000$.

$$\begin{aligned} &(\texttt{annual income} \geq 1000) \wedge (\texttt{seaside house} = \texttt{no}) && \text{(a)}\\ &(\texttt{annual income} \geq 2000) \wedge (\texttt{mountain house} = \texttt{no}) && \text{(b)}\\ &(\texttt{mountain house} = \texttt{yes}) \wedge (\texttt{seaside house} = \texttt{yes}) && \text{(c)} \end{aligned}$$

In a large set of edits, or in a phase of edits updating, inconsistencies may easily occur. Due to the following result, inconsistencies can be detected by solving a series of the satisfiability problems.

Theorem 3.1. *By encoding the set of edits in a CNF formula $\mathcal{E}$, complete inconsistency occurs if and only if $\mathcal{E}$ is unsatisfiable. A partial inconsistency with respect to a subset S_f^j occurs if and only if the formula obtained from $\mathcal{E}$ by fixing $\alpha_{S_f^j} = True$ is unsatisfiable.*

Moreover, in the case of inconsistency, we are interested in restoring consistency. The approach of deleting edits corresponding to clauses that we could not satisfy is not useful. In fact, every edit has its function, and cannot be deleted, but only modified by the human expert who writes the edits. On the contrary, the selection of the set of conflicting edits can guide the human expert in modifying them. This corresponds to selecting which part of the unsatisfiable CNF causes the unsolvability, i.e. a minimal unsatisfiable subformula (MUS). Therefore, we used a SAT solver which, in the case of unsatisfiable instances, is able to select a MUS or at least an unsatisfiable subformula approximating a MUS [4].

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

Example 3.3. A very simple redundancy, with edits meaning: (a) head of the house must have an annual income greater then or equal to 100, (b) everybody must have an annual income greater then or equal to 100. (a) is clearly redundant.

$$(\texttt{role} = \texttt{head of the house}) \wedge (\texttt{annual income} < 100) \quad \text{(a)}$$
$$\texttt{annual income} < 100 \quad \text{(b)}$$

A SAT formulation is used to solve the problem of logical implication. Given a set of statements S and a single statement s, $S \Rightarrow s$ if and only if $S \cup \neg s$ is an unsatisfiable formula [9] [10]. Therefore, the following holds.

Theorem 3.2. *The clausal representation of an edit e_j is implied by the clausal representation of a set of edits E if and only if $E \cup \neg e_j$ is unsatisfiable.*

It can be consequently checked if an edit with clausal representation e_j is redundant by testing if the formula $(\mathcal{E} \setminus e_j) \cup \neg e_j$ is unsatisfiable. Redundancy of every edit can be checked by applying to each one of them the above operation.

Detection of erroneous questionnaires Q^e trivially becomes the problem of checking if the truth assignment corresponding to Q satisfies the formula $\mathcal{E}$.

4 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. Given an *erroneous questionnaire* Q^e, the *imputation* process consists in changing some of his values, obtaining a *corrected questionnaire* Q^c which satisfies the formula $\mathcal{E}$ and is as close as possible to the (unknown) *original questionnaire* Q^o (the one we would have if we had no errors). Two general principles should be followed [6]: to apply the minimum changes to erroneous data, and to modify as less as possible the original frequency distribution of the data. Generally, a cost for changing each field is given, based on the reliability of the field. It is assumed that, when error is something unintentional, the erroneous fields are the minimum-cost set of fields that, if changed, can restore consistency.

The problem of *error localization* is to find a set W of fields of minimum total cost such that Q^c can be obtained from Q^e by changing (only and all) the values of W. Imputation of actual values of W can then be performed in a deterministic or probabilistic way. This cause the minimum changes to erroneous data, but has little respect for the original frequency distributions.

A *donor questionnaire* Q^d is a correct questionnaire which, according to some distance function $d(Q^e, Q^d) \in \mathbb{R}_+$, is the nearest one to the erroneous questionnaire Q^e, hence it represents a record with similar characteristics. The problem of *imputation trough a donor* is to find a set D of fields of minimum total cost such that Q^c can be obtained from Q^e by copying from the donor Q^d (only and all) the values of D. This is generally recognized to cause low alteration of the original frequency distributions, although changes caused to erroneous data are not minimum. We are interested in solving both of the above problems.

Example 4.1. We have the following erroneous questionnaire Q^e

$\{\ldots\ \texttt{age} = \texttt{17}, \texttt{car} = \texttt{no}, \texttt{city of residence} = \texttt{A}, \texttt{city of work} = \texttt{B}\ \ldots\}$

where the solution of error localization is $W = \{$ `city of work` $\}$, with a total cost of 2.5 (suppose in fact we could restore consistency if we have `city of work` = `A`). Searching for a donor, however, we have Q^d such that

{... `age` = `18`, `car` = `yes`, `city of residence` = `A`, `city of work` = `B` ...}

The solution of imputation through a donor is $D=$ { `age`, `car` }, with a total cost of $c = 3$. By copying D from the donor, we obtain Q^c:

{... `age` = `18`, `car` = `yes`, `city of residence` = `A`, `city of work` = `B` ...}

5 A Set Covering Formulation

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 [11] is the problem of taking the set of elements s_i of minimum total weight such that at least one element for every A_j is taken. Let a^j be the incidence vector of A_j, i.e. a vector in $\{0,1\}^n$ whose i-th component a^j_i is 1 if $s_i \in A_j$, 0 if $s_i \notin A_j$. By using a vector of variables $x \in \{0,1\}^n$ which is the incidence vector of the set of taken elements s_i, we have

$$\begin{aligned} &\min \sum_{i=1}^{n} c_i x_i \\ &\text{s. t.} \sum_{i=1}^{n} a^j_i x_i \geq 1 \quad j = 1 \dots m, \qquad x \in \{0,1\}^n \end{aligned} \tag{2}$$

This problem is well-known NP-complete [7], and is of great relevance in many applied fields. In order to work with binary optimization, a positive literal α_i becomes a binary variable x_i, and a negative literal $\neg\alpha_i$ becomes a negated binary variable $\bar{x}_i$. A questionnaire Q, which mapped to a truth assignment in $\{True, False\}^n$, will now map to a binary vector in $\{0,1\}^n$. A clause c_j

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

becomes now the following linear inequality, by defining the set A_π of the logical variables appearing positive in c_j, and the set A_ν of the logical variables appearing negated in c_j, and the corresponding incidence vectors a^π and a^ν

$$\sum_{i=1}^{n} a^\pi_i x_i + \sum_{i=1}^{n} a^\nu_i \bar{x}_i \geq 1$$

Example 5.1. Suppose that the (correct) questionnaire Q maps to the truth assignment $\{\alpha_1 = False, \alpha_2 = False, \alpha_3 = True\}$, and that $\mathcal{E}$ 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 Q is $\{x_1 = 0, x_2 = 0, x_3 = 1\}$, and the system of linear inequalities corresponding to $\mathcal{E}$ is

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

We can now model the two above imputation problems as follows. We have

- The binary vector $e = \{e_1, \ldots, e_n\} \in \{0,1\}^n$ corresponding to the erroneous questionnaire Q^e.
- In the case of *imputation through a donor* only, 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 $x_i + \bar{x}_i = 1$. They correspond 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 furthermore introduce a vector of binary variables $y = \{y_1, \ldots, y_n\} \in \{0,1\}^n$ representing the changes we introduce in e.

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

The minimization of the total cost of the changes can be expressed with

$$\min_{y_i \in \{0,1\}} \sum_{i=1}^{n} c_i y_i \quad = \quad \min_{y \in \{0,1\}^n} c'y \tag{3}$$

However, the constraints are expressed for x. A key issue is that there is a relation between variables y and x (and consequently $\bar{x}$). In the case of error localization, 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 the case of imputation trough a donor, this depends on the values of e and d.

$$y_i = \begin{cases} x_i \quad (= 1 - \bar{x}_i) & \text{if } \ e_i = 0 \text{ and } d_i = 1 \\ 1 - x_i \quad (= \bar{x}_i) & \text{if } \ e_i = 1 \text{ and } d_i = 0 \\ 0 & \text{if } \ e_i = d_i \end{cases} \tag{5}$$

By using the above results, we can express the two imputation problems with the following set covering formulation. In the case of error localization, (3) becomes

$$\min_{x_i, \bar{x}_i \in \{0,1\}} \ \sum_{i=1}^{n} (1 - e_i) c_i x_i + \sum_{i=1}^{n} e_i c_i \bar{x}_i \tag{6}$$

Conversely, in the case of imputation through a donor, our objective (3) becomes

$$\min_{x_i, \bar{x}_i \in \{0,1\}} \ \sum_{i=1}^{n} (1 - e_i) d_i c_i x_i + \sum_{i=1}^{n} e_i (1 - d_i) c_i \bar{x}_i \tag{7}$$

Subject, in both cases (6) and (7), to the following set of constraints

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

6 Implementation and Results

Satisfiability problems are solved by means of the efficient enumerative solver Adaptive Core Search (ACS) [4]. In the case of unsatisfiable instances, ACS is able to select a subset of clauses which are still unsatisfiable. Set covering are solved by means of a solver based on the Volume Algorithm [2]. This effective procedure recently presented by Barahona is an extension of the subgradient algorithm, wich is able to produce primal as well as dual solutions. We added a simple heuristic in order to obtain an integer solution to our set covering problem. Moreover, since this is an approximate procedure, we compared its results with the commercial branch-and-bound solver *Xpress*. The process of edits validation and data imputation in the case of a Census of Population is performed. Edits are provided by the Italian National Statistic Institute (Istat). We checked different sets of real edits, and (in order to test the limits of the procedure) artificially generated CNF instances of larger size (up to 15000 var. and 75000 clauses) representing simulated sets of edits. When performing the whole inconsistency and redundancy checking, every CNF with n variables and m clauses, produces about $1 + n + n/10 + m$ SAT problems of non trivial size. Total times for sequentially solving all such SAT problems (on a Pentium II 450MHz PC) are reported. Inconsistency or redundancy present in the set of edits were detected in the totality of the cases.

Real sets of edits			
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

Simulated sets of edits			
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

Tables 1 & 2: Edit validation procedure on real and simulated sets of edits.

Real problems 480 var. and 1880 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

Simulated problems 30000 var. and 90000 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	-

Tables 3 & 4: Error localization procedure on real and simulated sets of edits.

Real problems 480 var. and 1880 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

Simulated problems 30000 var. and 90000 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

Tables 5 & 6: Imputation through a donor on real and simulated sets of edits.

Detection of erroneous questionnaires was performed, as a trivial task. In the cases of error localization and imputation through a donor, for each set of edits

we considered various simulated erroneous answers with different percentage of activated edits. Note that a CNF with n variables and m clauses corresponds to a set covering problem with $2n$ variables and $m + n$ constraints. In the case of error localization, the heuristic VA can solve problems of size not solvable by $B\&B$ within the time limit of 6 hours. Further occasional tests with higher error percentage shows that VA does not increase its running time, but it is often unable to find a feasible integer solution, while B&B would reach such solution but in an a prohibitive amount of time. In the case of imputation through a donor, the heuristic VA is unable to find a feasible integer solution for problems with large error percentage, while $B\&B$ solves all problems in very short times. This holds because, in the case of the use of a donor, many variables are fixed if $e_i = d_i$ (see (5)). Therefore, such problems become similar to error localization problems with a smaller number of variables but a higher error percentage.

7 Conclusions

A binary encoding is the more direct and effective representation both for records and for the set of edit rules, allowing automatic detection of 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. 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.

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. E. Boros, P.L. Hammer, T. Ibaraki and A. Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.
4. 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, 500–505, 2000.
5. R. Bruni and A. Sassano. CLAS: a Complete Learning Algorithm for Satisfiability. *Dip. di Inf. e Sist., Univ. di Roma "La Sapienza"* , Technical Report 01-01, 1999.
6. P. Fellegi and D. Holt. A Systematic Approach to Automatic edit and Imputation. *Journal of the American Statistical Association*, 17:35–71(353), 1976.
7. 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.
8. M.R. Genesereth and N.J. Nillson. *Logical Foundation of Artificial Intelligence.* Morgan Kaufmann, 1987.
9. R. Kowalski. *Logic for Problem solving.* North Holland, 1978.
10. D.W.Loveland. *Automated Theorem Proving: a Logical Basis.* North Holland 1978.
11. G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* J. Wiley, New York, 1988.
12. 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.
13. W.E. Winkler. State of Statistical Data Editing and current Research Problems. *UN/ECE Work Session on Stat. Data Edit.*, W. P. n.29, Rome, Italy, 2-4 June 1999.