Error Correction for Massive Data Sets*

Renato Bruni
Università di Roma "La Sapienza" - D.I.S.
Via M. Buonarroti 12, Roma, Italy, 00185.
E-mail: bruni@dis.uniroma1.it

September 16, 2003

Abstract

The paper is concerned with the problem of automatic detection and correction of errors into massive data sets. As customary, erroneous data records are detected by formulating a set of rules. Such rules are here encoded into linear inequalities. This allows to check the set of rules for inconsistencies and redundancies by using a polyhedral mathematics approach. Moreover, it allows to correct erroneous data records by introducing the minimum changes through an integer linear programming approach. Results of a particularization of the proposed procedure to a real-world case of census data correction are reported.

Keywords: Data correction, Inconsistency localization, Massive data sets.

1 Introduction

In the past, for several fields, an automatic information processing have often been prevented by the scarcity of available *data*. Nowadays data have become more and more abundant, but the problem that frequently arises is that such data may contain *errors*. This again makes an automatic processing not applicable, since the result is not reliable. Data *correctness* is indeed a crucial aspect of data quality. The relevant problems of error *detection* and *correction* should therefore be solved. When dealing with *massive* data sets, such problems are particularly difficult to formalize and very computationally demanding to solve.

As customary for structured information, data are organized into *records*. A record has the formal structure of a set of *fields*, or *attributes*. Giving a *value* to each field, a record instance, or, simply, a record, is obtained [29]. The problem of error detection is generally approached by formulating a set of *rules* that each

*Work developed during the research collaboration between the University of Roma "La Sapienza" and the Italian Statistic Office (Istat).

record must respect in order to be declared *correct*. A record not respecting all the rules is declared *erroneous*. In the field of database theory, rules are also called *integrity constraints* [29], whereas in the field of statistics, rules are also called *edits* [13], and express the error condition. The automatic extraction of such rules, or in general of patterns, from a given data-set, constitutes a central problem for the areas of data mining and data analysis, particularly in their description aspects [5, 12, 22, 23].

The problem of *error correction* is usually tackled by changing some of the values of an erroneous record, in order to obtain a *corrected record* which satisfies the above rules but preserves as much as possible the information contained in the erroneous record. This is deemed to produce a record which should be as close as possible to the (unknown) *original record* (the record that would be present in absence of errors). Because of its relevance and spread, the above problem have been extensively studied in a variety of scientific communities. In the field of statistics, the correction process is often subdivided into an *error localization* phase, in which the set of erroneous values within each record is determined, and a *data imputation* phase, in which the correct values for the erroneous fields are imputed. Several different rules encoding and solution algorithm have been proposed (e.g. [3, 11, 27, 35]). A very well-known approach to the problem, which implies the generation of all rules logically implied by the initial set of rules, is due to Fellegi and Holt [13]. In practical case, however, such methods suffer from severe computational limitations [27, 35], with consequent heavy limitations on the number of rules and records that can be considered. In the field of computer science, on the other hand, the correction process is also called *data cleaning*. Such problem is tackled by extracting patterns from data in [19]. Errors may also be detected as inconsistencies in knowledge representation, and corrected with many consistency restoring techniques [2, 25, 30]. Another approach to error correction, in database theory, consists in performing a cross validation among multiple sources of the same information [29]. This involves relevant *linkage* problems. Recently, a declarative semantics for the imputation problem has been proposed in [14], as an attempt to give an unambiguous formalization of the meaning of imputation.

In particular, previous approaches to data correction problems by using mathematical programming techniques have been used. By requiring to change at least one of the values involved in each violated rule, a (mainly) *set covering* model of the error localization problem have been considered by many authors. Such model have been solved by means of *cutting plane* algorithms in [15] for the case of *categorical* data, and in [16, 28] for the case of *continuous* data. The above procedures have been adapted to the case of a mix of categorical and continuous data in [11], were a *branch-and-bound* approach to the problem is also considered. Such model, however, does not represent all the problem's features, in the sense that the solution to such model may fail to be a solution to the localization problem, the separation of the error localization phase from the imputation phase may originate artificial restrictions during the latter one, and computational limitations still hold.

An automatic procedure for generic data correction by using new discrete

mathematical models of the problem is here presented. A similar but more limited procedure, which uses only a propositional logic encoding, was described in an earlier paper [7]. Our rules, obtained from several sources (e.g. human expertise, machine learning), are accepted according to a specific syntax and automatically encoded into linear inequalities, as explained in Sect. 2. Such set of rules should obviously be free from *inconsistencies* (i.e. rules contradicting other rules) and, preferably, from *redundancies* (i.e. rules which are logically implied by other rules). As a first relevant point, the set of rules itself is checked for inconsistencies and redundancies by using the polyhedral mathematics approaches shown in Sect. 3. Inconsistencies are selected by selecting *irreducible infeasible subsystems* (IIS, see also [1, 9, 10, 32]) by using a variant of Farkas' lemma (see e.g. [31]), while redundancies are detected by finding implied inequalities. This allows the use of a set of rules much more numerous than other procedures, with consequent increased detection power, and allows moreover a much easier merging and updating of existing sets of rules, with consequent increased flexibility. After this validation phase, rules are used to detect erroneous records. Such records are then corrected, hence changed in order to satisfy the above rules. As a second relevant point, the correction problem is modeled as the problem of minimizing the weighted sum of the changes subject to constraints given by the rules, by using the integer programming formulations described in Sect. 4. This allows the use of very efficient solution techniques, besides of having the maximum degree of flexibility with respect to data meaning. The proposed procedure is tested by executing the process of error detection and correction in the case of real world census data, as clarified in Sect. 5. The above depicted models are solved by means of a state-of-the-art integer programming solver (ILOG Cplex [21]). The practical behavior of the proposed procedure is evaluated both from the computational and from the data quality point of view. The latter analysis is carried out by means of recognized statistical indicators [24]. The overall software system developed for the census application, called DIESIS (Data Imputation Editing System - Italian Software) is described in [6].

2 Encoding Rules into Linear Inequalities

In Database theory, a *record schema* is a set of fields f_i, with $i = 1 \dots m$, and a *record instance* is a set of values v_i, one for each of the above fields. In order to help exposition, we will focus on records representing *persons*. Note, however, that the proposed procedure is completely general, because it is not influenced by the meaning of processed data. The record scheme will be denoted by P, whereas a generic record instance corresponding to P will be denoted by p.

$$P = \{f_1, \dots, f_m\} \qquad p = \{v_1, \dots, v_m\}$$

Example 2.1. For records representing persons, fields are for instance `age` or `marital status`, and corresponding examples of values are `18` or `single`.

Each field f_i, with $i = 1 \dots m$, has its *domain* D_i, which is the set of every pos-

sible value for that field. Since we are dealing with errors, the domains include all values that can be found in data, even the erroneous ones. A distinction is usually made between *quantitative*, or *numerical*, fields, and *qualitative*, or *categorical* fields. A quantitative field is a field on whose values are applied (at least some) mathematical operators (e.g. $>$, $+$), hence such operators should be defined on its domain. Examples of quantitative fields are numbers, or even the elements of an ordered set. Quantitative fields can be either *continuous* (e.g. real numbers) or *discrete* (e.g. integer numbers) ones. A qualitative field simply requires its domain to be a discrete set with finite number of elements. We are not interested here in considering fields ranging over domains having a non-finite number of non-ordered values. The proposed approach is able to deal with both qualitative and quantitative values.

Example 2.2. 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 (`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

$$D_{\texttt{age}} = \mathbb{Z} \cup \{\texttt{blank}\}$$

A record instance, and in particular a person instance p, is declared correct if and only if it respects a *set of rules* denoted by $R = \{r_1, \ldots, r_u\}$. Each rule can be seen as a mathematical function r_k from the Cartesian product of all the domains to the Boolean set {0,1}, as follows.

$$\begin{array}{rccc} r_k: & D_1 \times \ldots \times D_m & \rightarrow & \{0,1\} \\ & p & \mapsto & 0,1 \end{array}$$

Rules are such that p is a correct record if and only if $r_k(p) = 1$ for all $k = 1 \ldots u$. Rules should be expressed according to some syntax. In our case, each rule is expressed as a disjunction ($\vee$) of elementary statements called *conditions* (α_h). Conditions can also be negated ($\neg\alpha_h$). Therefore, rules have the structure of clauses (which are disjunctions of possibly negated propositions). By introducing, for each rule r_k, the set π_k of the indices of its positive conditions and the set ν_k of the indices of its negative conditions, r_k can be written as follows.

$$\bigvee_{h \in \pi_k} \alpha_h \vee \bigvee_{h \in \nu_k} \neg\alpha_h \tag{1}$$

Since all rules must be respected, a conjunction ($\wedge$) of conditions is simply expressed using a set of different rules, each made of a single condition. As known, all other logic relations between conditions (implication $\Rightarrow$, etc.) can be expressed by using only the above operators ($\vee$, $\wedge$, $\neg$). Conditions have here an internal structure, and we need to distinguish between two different kind of

structures. A condition involving values of a single field is here called a *logical condition*. A condition involving mathematical operations between values of fields is here called *mathematical condition*.

Example 2.3. A logical condition can be, for instance, $(\texttt{age} < 14)$, or $(\texttt{marital status} = \texttt{married})$. A mathematical conditions can be, for instance: $(\texttt{age - years married} \geq 14)$.

We call *logical rules* the rules expressed only with logical conditions, *mathematical rules* the rules expressed only with mathematical conditions, and *logic-mathematical rules* the rules expressed using both type of conditions.

Very often, some values of a domain D_i are not acceptable, regardless of values of all other fields. Such values are called *out-of-range* values. By removing the out-of-range values from a domain D_i, we obtain the *feasible domain* $\mathring{D}_i \subseteq D_i$. Feasible domains are delimited by using logical rules.

Example 2.4. A logical rule expressing that all people declaring to be married should be at least 14 years old is:

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

Logical rules delimiting the feasible domain for the field `age` are for instance:

$$(\texttt{age} \geq 0),\ (\texttt{age} \leq 110)$$

One can observe that small modifications of some value v_i may have no influence on the value returned by a rule r_k. Formally, we say that two values v'_i and v''_i are *equivalent* with respect to rule r_k and write $v'_i \overset{r_k}{\simeq} v''_i$ when, for every couple of records p' and p'' having all values identical except for field f_i, as below, r_k has the same value on p' and p'':

$$v'_i \overset{r_k}{\simeq} v''_i \iff r_k(\{v_1, \ldots, v'_i, \ldots, v_m\}) = r_k(\{v_1, \ldots, v''_i, \ldots, v_m\})$$

In other words, when considering only rule r_k, records p' and p'' are either both correct or both erroneous. It is easy to see that the above is an equivalence relationship, since it is reflexive ($v'_i \overset{r_k}{\simeq} v'_i$), symmetrical ($v'_i \overset{r_k}{\simeq} v''_i \Rightarrow v''_i \overset{r_k}{\simeq} v'_i$) and transitive ($v'_i \overset{r_k}{\simeq} v''_i$ and $v''_i \overset{r_k}{\simeq} v'''_i \Rightarrow v'_i \overset{r_k}{\simeq} v'''_i$). A straightforward generalization is the notion of equivalence with respect to a *set* of rules. A key point is the following lemma.

Lemma 2.1. *Each domain D_i can always be partitioned into n_i subsets*

$$D_i = S_{i\,1} \cup \ldots \cup S_{i\,n_i}$$

in such a way that all values belonging to the same S_{ij} are equivalent with respect to the logical rules (i.e. considering all and only the logical rules). Such subsets are the equivalence classes for the equivalence relationship introduced.

Such partition is obtained as follows. Values of the domains explicitly appearing in the rules are called *breakpoints*, or *cut-points*, for the domains. All breakpoints concerning domain D_i represent logical *watersheds* among the values of D_i. Their set will be denoted by B_i, while the single breakpoints by β_{ij}. We have

$$B_i = \{\beta_{i\,1}, \ldots, \beta_{i\,n_i'}\}$$

Domain D_i can now be *cut* in correspondence of each breakpoint in order to obtain subsets (which are intervals for continuous fields, sequences of values for discrete fields, sets of values for qualitative fields). Note that n_i' may be different from n_i. By furthermore merging possibly equivalent subsets, we obtain the above mentioned n_i subsets $\{S_{i\,1}, \ldots, S_{i\,n_i}\}$ for each field f_i. A subset for the `out-of-range` values is always present. Moreover, the value for some field can be the missing value. Such value is called `blank`, or `null` and, depending on the field, can belong or not to the feasible domain. Typically, the blank value belongs to a feasible domain $\mathring{D}_i$ when there exist situations in which the value of field f_i becomes non-existent (e.g. date of marriage for unmarried people). If the blank answer belongs to the feasible domain a subset for `blank` is also used. Otherwise, the blank answer belongs to the `out-of-range` subset.

In all the examples, subsets S_{ij} will be denoted by simply writing the name of the field (e.g. `age`) as the field index i, and the condition (e.g. $\in \{0...13\}$, etc.) as the subset index j.

Example 2.5. Suppose that, by scanning a given set of rules R, the following set of breakpoints $B_{\texttt{age}}$ is obtained for the field `age` of a person.

$$B_{\texttt{age}} = \{\texttt{0},\ \texttt{14},\ \texttt{18},\ \texttt{26},\ \texttt{110},\ \texttt{blank}\}$$

Therefore, by using $B_{\texttt{age}}$ and R, the following subsets are obtained. The last subset is the out-of-range one.

$$S_{\texttt{age}\in\{0...13\}} = \{\texttt{0}, \ldots, \texttt{13}\},\ S_{\texttt{age}\in\{14...17\}} = \{\texttt{14}, \ldots, \texttt{17}\},$$
$$S_{\texttt{age}\in\{18...25\}} = \{\texttt{18}, \ldots, \texttt{25}\},\ S_{\texttt{age}\in\{26...110\}} = \{\texttt{26}, \ldots, \texttt{110}\},$$
$$S_{\texttt{age = out\_of\_range}} = \{\ldots, -\texttt{1}\} \cup \{\texttt{111}, \ldots\} \cup \{\texttt{blank}\}$$

Now, the *variables* for the announced linear inequalities can be introduced: a set of m integer variables $z_i \in \{0, \ldots, U\}$, one for each domain D_i, and a set of $n = n_1 + \ldots + n_m$ binary variables $x_{ij} \in \{0, 1\}$, one for each subset S_{ij}. We represent each value v_i of p with an integer variable z_i, by defining a mapping φ between values of the domain and integer numbers between 0 and an upper value U. U is the same for all fields, and is such that no elements of any feasible domain maps to U.

$$\begin{array}{rccc} \varphi_i: & D_i & \rightarrow & \{0, \ldots, U\} \\ & v_i & \mapsto & z_i \end{array}$$

Mapping for integer and rational domains is straightforward. We approximate real domains with rational domains (there is no difference in computer representation) and then map them on the set of integer numbers $\{0, \ldots, U\}$. Qualitative

domains also are mapped on $\{0, \ldots, U\}$ by choosing an arbitrary ordering. Note that, in the case of the considered application, values were wanted to be integer. However, variables z_i are not structurally forced to be integer, and the proposed models are easily modifiable in case of z_i being continuous variables. All the `out-of-range` values map to the greater number used U. The `blank` value, when belonging to the feasible domain, maps to an integer value η_i immediately consecutive to the greater numerical value of the feasible domain $\mathring{D}_i$. Note that $\eta_i < U$ is always required.

Example 2.6. For the field `years married`, if the domain is $\mathbb{Z} \cup \{\texttt{blank}\}$ and the feasible domain is $\{\texttt{0}, ..., \texttt{91}\} \cup \{\texttt{blank}\}$, the mapping $\varphi_{\texttt{years married}}$ is

$$\begin{array}{rccccccc} v_{\texttt{years married}} = & \underbrace{\ldots, -1} & \texttt{0} & \ldots & \texttt{91} & \texttt{blank} & \underbrace{92, \ldots} \\ & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ z_{\texttt{years married}} = & U & 0 & \ldots & 91 & 92 & U \end{array}$$

The membership of a value v_i to the subset S_{ij} is encoded by using the binary variables x_{ij}.

$$x_{ij} = \begin{cases} 1 & \text{when} \quad v_i \in S_{ij} \\ 0 & \text{when} \quad v_i \notin S_{ij} \end{cases}$$

Integer and binary variables are linked by using a set of linear inequalities called *bridge constraints*. They impose that, when z_i has a value such that v_i belongs to subset S_{ij}, the corresponding x_{ij} is 1 and all others binary variables $\{x_{i1}, \ldots, x_{ij-1}, x_{ij+1}, \ldots, x_{in_i}\}$ of field f_i are 0.

By using these variables, all the above rules can be expressed. A logic condition α_h identifies a part of a certain domain D_i, hence it corresponds to a certain subset S_{ij} (or to a union of such subsets). Therefore, each logic condition α_h is simply substituted by the corresponding binary variable x_{ij} (or by a sum of such variables), while each mathematical condition α_h is expressed by using the integer variables z_i. By doing so, each logical rule r_k having the structure (1) of a clause can be written as the following linear inequality

$$\sum_{(i,j)\in\pi_k} x_{ij} + \sum_{(i,j)\in\nu_k} (1 - x_{ij}) \geq 1$$

When mathematical conditions are present, the only difference is that they do not correspond to binary variables but to operations between the integer variables. In order to obtain linear inequalities, we limit mathematical rules to those which are linear or linearizable. For a digression on linearizable inequalities, see for instance [34]. Occasionally, further binary variables are introduced, for instance to encode disjunctions of mathematical conditions. Note, moreover, that a very precise syntax for rules was developed for the case described in [6]. Therefore, encoding could be performed by means of an automatic procedure, which reads the list of the fields and the rules written using such syntax, determines all the breakpoints β_{ij} and consequently the subsets S_{ij}, generates the

suitable variables x_{ij} and z_i, re-reads each rule and generates the corresponding linear inequality (or inequalities) as in the following example.

Example 2.7. Consider the following logical rule introduced in Example 2.4.

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

By substituting the logical conditions, it becomes the linear inequality:

$$(1 - x_{\texttt{marital\_status = married}}) + (1 - x_{\texttt{age}\in\{0\ldots13\}}) \geq 1$$

Consider, instead, the following logic-mathematical rule.

$$\neg(\texttt{marital status} = \texttt{married}) \vee (\texttt{age} - \texttt{years married} \geq 14)$$

By substituting the logical and mathematical conditions, we have

$$(1 - x_{\texttt{marital status = married}}) \vee (z_{\texttt{age}} - z_{\texttt{years married}} \geq 14)$$

which becomes the following linear inequality

$$U(1 - x_{\texttt{marital status = married}}) + z_{\texttt{age}} - z_{\texttt{years married}} \geq 14$$

Altogether, from the set of rules R, a set of linear inequalities is obtained. Each record p determines an assignment of values for the introduced variables x_{ij} and z_i. By construction, all and only the variable assignments corresponding to correct records satisfy all the linear inequalities. By denoting with x and z the vectors respectively made of all the components x_{ij} and z_i, $i = 1 \ldots m$, $j = 1 \ldots n_i$, as follows,

$$x = (x_{11}, \ldots, x_{1n_1}, \ldots, x_{m1}, \ldots, x_{mn_m})^T \quad z = (z_1, \ldots, z_m)^T$$

the set of rules R becomes a system of linear inequalities, expressed in compact notation as follows.

$$\begin{cases} B \begin{bmatrix} x \\ z \end{bmatrix} \geq b \\ x \in \{0,1\}^n \\ z \in \{0, \ldots, U\}^m \end{cases} \tag{2}$$

Since x has $n = n_1 + \ldots + n_m$ components and z has m components, and letting l be the total number of inequalities, B is in general a $l \times (n + m)$ real matrix, and b a real l-vector. Briefly, even if slightly improperly, a record p must satisfy (2) to be correct.

3 Validation of the Set of Rules

However, due to several reasons, the set of rules R may be affected by the presence of inconsistencies or redundancies. When every possible record p is incorrectly declared erroneous because of a rule inconsistency, we have the situation called *complete inconsistency* of the set of rules. When the rules' inconsistency appears only for particular values of particular fields, we have the (even more insidious) situation of *partial inconsistency*.

Example 3.1. A very simple complete inconsistency, with rules 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. Note that more complex inconsistencies are not so easily visible.

$$\begin{cases} x_{\texttt{seaside house = yes}} \geq 1 & \text{(a)} \\ x_{\texttt{mountain house = yes}} \geq 1 & \text{(b)} \\ (1 - x_{\texttt{seaside house = yes}}) + (1 - x_{\texttt{mountain house = yes}}) \geq 1 & \text{(c)} \end{cases}$$

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

$$\begin{cases} (1 - x_{\texttt{annual income} \geq \texttt{1000}}) + (x_{\texttt{seaside house = yes}}) \geq 1 & \text{(a)} \\ (1 - x_{\texttt{annual income} \geq \texttt{2000}}) + (x_{\texttt{mountain house = yes}}) \geq 1 & \text{(b)} \\ (1 - x_{\texttt{seaside house = yes}}) + (1 - x_{\texttt{mountain house = yes}}) \geq 1 & \text{(c)} \end{cases}$$

In large sets of rules, or after rule updating, inconsistencies may easily occur. Such inconsistencies corresponds to structural properties of the system (2).

Theorem 3.1. *By encoding the set of rules as the system of linear inequalities (2), a complete inconsistency occurs if and only if (2) is infeasible, i.e. has no integer solutions. A partial inconsistency with respect to a subset S_{ij} occurs if and only if the system obtained by adding $x_{ij} = 1$ to (2) is infeasible, i.e. has no integer solutions.*

Proof: In the first case, by definition of complete inconsistency, the set of rules does not admit any correct record. Therefore, the corresponding system of linear inequalities (2) does not admit any solution, hence is infeasible. In the second case, instead, by definition of partial inconsistency, every record having a value belonging to a certain subset $S_{ij}^{\dagger}$ is declared erroneous. Therefore, the corresponding system of linear inequalities (2) does not admit any solution with $x_{ij}^{\dagger} = 1$. Hence, by adding such constraint to (2), we obtain an infeasible system.

Moreover, in the case of inconsistency, we are interested in restoring consistency. The approach of deleting rules corresponding to inequalities that could not be satisfied is not useful. This because every specific rule allows to detect a specific data error, and cannot be deleted. On the contrary, all the *conflicting rules* should be located. The selection of the set of conflicting rules guides the rule repair process, during which such rules are modified by their original source (typically the human expert who writes the rules). Postinfeasibility analysis, in fact, "requires the cooperation of *algorithmic engine* and *human intelligence*" [10]. The selection of such set of conflicting rules corresponds to selecting a part of system (2) causing the infeasibility. An *irreducible infeasible subsystem* (IIS) is a subset of the inequalities of an infeasible system that is itself infeasible, but for which any proper subset is feasible. Note that, since we are interested in integer solutions, we actually look for an *integer* IIS, that is an irreducible subset of inequalities having no integer solutions [18]. Therefore, checking the rules for inconsistencies produces a series of integer IIS selection problems.

In the case of systems of linear inequalities were we are interested in real-valued solutions, the following result on infeasibility holds:

Theorem 3.2 (Farkas' lemma) *Let A be an $s \times t$ real matrix and let a be a real s-vector. Then there exists a real t-vector $x \geq \mathbf{0}$ with $Ax = a$ if and only if $y^T a \geq 0$ for each real s-vector y with $y^T A \geq \mathbf{0}$.*

A proof is for instance in [31]. Geometrically, this means that if an s-vector γ does not belong to the cone generated by the s-vectors $a_1, \ldots, a_t$ (columns of A), there exists a linear hyperplane separating γ from $a_1, \ldots, a_t$. There are several equivalent forms of Farkas' lemma. The following variant is more suitable to our purposes. Given a matrix $A \in \mathbb{R}^{s \times t}$ and a vector $a \in \mathbb{R}^s$, consider the system:

$$\begin{cases} Ax & \leq & a \\ x & \in & \mathbb{R}^t \end{cases} \tag{3}$$

and the new system of linear inequalities obtained from the former one:

$$\begin{cases} y^T A & = & \mathbf{0} \\ y^T a & < & 0 \\ y & \geq & \mathbf{0} \\ y & \in & \mathbb{R}^s \end{cases} \tag{4}$$

We have that exactly one of the two following possibilities holds:

- (1) is feasible, i.e. there exists $x \in \mathbb{R}^t$ verifying all its inequalities.
- (2) is feasible, i.e. there exists $y \in \mathbb{R}^s$ verifying all its inequalities.

An IIS can be selected within (3) by solving the following new system [17]:

$$\begin{cases} y^T A & = & \mathbf{0} \\ y^T a & \leq & -1 \\ y & \geq & \mathbf{0} \\ y & \in & \mathbb{R}^s \end{cases} \tag{5}$$

The *support* of a vertex denotes the indices of its non-zero components; $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{U}$ respectively denote vectors of zeroes, ones and Us of appropriate dimension.

Theorem 3.3. (Gleeson and Ryan) *Consider two systems of linear inequalities respectively in form (3) and (5). If (5) is infeasible, (3) is feasible. On the contrary, if (5) is feasible, (3) is infeasible, and, moreover, each IIS of (3) is given by the support of each vertex of the polyhedron (5).*

The proof is based on polyhedral arguments using properties of extreme rays, see [17]. Therefore, checking the feasibility of (3), and, if infeasible, identifying one of its IIS, becomes the problem of finding a vertex of a polyhedron.

However, in the case of (2), we have a systems of linear inequalities were we are interested in integer solutions. In order to use the results given for the linear case, let us consider the linear relaxation of such system (2).

$$\begin{cases} -B \begin{bmatrix} x \\ z \end{bmatrix} & \leq & -b \\ \begin{bmatrix} x \\ z \end{bmatrix} & \leq & \begin{bmatrix} \mathbf{1} \\ \mathbf{U} \end{bmatrix} \\ -\begin{bmatrix} x \\ z \end{bmatrix} & \leq & \mathbf{0} \\ \begin{bmatrix} x \\ z \end{bmatrix} & \in & \mathbb{R}^{n+m} \end{cases} \tag{6}$$

The above system (6) is now in the form of (3). The l inequalities from the first group will be called *rules inequalities*, even if, for some of them, there can be no one-to-one correspondence with rules (see Sect. 2). By denoting with I the identity matrix, the $[l + 2(n + m)] \times (n + m)$ matrix A and the $[l + 2(n + m)]$-vector a are composed as follows. Number of rows for each block is reported on the left.

$$A = \begin{bmatrix} -B \\ I \\ -I \end{bmatrix} \begin{matrix} l \\ n+m \\ n+m \end{matrix} \qquad a = \begin{bmatrix} -b \\ \mathbf{1} \\ \mathbf{U} \\ \mathbf{0} \end{bmatrix} \begin{matrix} l \\ n \\ m \\ n+m \end{matrix}$$

Therefore, a system which plays the role of (3) can now be written.

$$\begin{cases} y^T \begin{bmatrix} -B \\ I \\ -I \end{bmatrix} & = & \mathbf{0} \\ y^T \begin{bmatrix} -b \\ \mathbf{1} \\ \mathbf{U} \\ \mathbf{0} \end{bmatrix} & \leq & -1 \\ y \geq \mathbf{0}, \quad y \in \mathbb{R}^{[l+2(n+m)]} \end{cases} \tag{7}$$

So far, the following result on the pair of systems (2) and (7) holds. The *restriction* of the support of a vertex to rules inequalities will denote the indices of its non-zero components among those corresponding to rules inequalities.

Theorem 3.4. *Consider two systems of linear inequalities respectively in form (2) and (7). In this case, if (7) is feasible, (2) is infeasible, and the restriction of the support of each vertex of the polyhedron (7) to rules inequalities contains an integer IIS of (2). On the contrary, if (7) is infeasible, (6) is feasible, but it cannot be decided whether (2) is feasible or not.*

Proof: We first prove that the restriction of the support of a vertex of (7) to rule inequalities contains an integer IIS of (2). Assume (7) is feasible, and let v_1 be the vertex found. Therefore, (6) is infeasible (from Theorem 3.2), and an IIS in (6), called here IIS_1, is given by the support of v_1. Such IIS_1 is in general composed by a set RI_1 of *rules inequalities* and a set BC_1 (possibly empty) of *box constraints* (the ones imposing $0 \leq x_{ij} \leq 1, 0 \leq z_i \leq U$). The set of inequalities RI_1 has no integer solutions, since removing the BC_1 from IIS_1, while imposing the more strict *integer constraints* IC_1 (the ones imposing $x_{ij} \in \{0,1\}, z_i \in \{0,...,U\}$), keeps IIS_1 unsatisfiable. Therefore, an integer IIS is contained into RI_1. The integer IIS may also be a subset of the inequalities of RI_1, because, though $IIS_1 = RI_1 \cup BC_1$ is minimally infeasible, $RI_1 \cup IC_1$ may be not minimal: we are imposing the more strict integer constraints instead of the box constraints. Therefore, the procedure produces an integrally infeasible subsystem containing an integer IIS for (2).

On the other hand, not all integer IIS in (2) can be obtained by such procedure. This because, if (7) is infeasible, (6) is feasible (by Theorem 3.2). When imposing the more strict integer constraints instead of the box constraints, however, nothing can be said on the feasibility of (2).

Example 3.3. Consider a set of rules R on two conditions α_1, α_2, as follows. One may not note that R contains an inconsistency of the type of Example 3.1.

$$r_1 = (\alpha_1),\ r_2 = (\alpha_2),\ r_3 = (\neg\alpha_1 \vee \neg\alpha_2),\ r_4 = (\alpha_1 \vee \neg\alpha_2)$$

In this case, $n = 2$ and m can be considered 0, since no z variables are needed to express the above rules. A and a can easily be obtained, as follows.

$$A = \left[\begin{array}{rr} -1 & 0 \\ 0 & -1 \\ 1 & 1 \\ -1 & 1 \\ \hline 1 & 0 \\ 0 & 1 \\ \hline -1 & 0 \\ 0 & -1 \end{array}\right] \qquad a = \left[\begin{array}{r} -1 \\ -1 \\ 1 \\ 0 \\ \hline 1 \\ 1 \\ \hline 0 \\ 0 \end{array}\right]$$

Therefore, the system to be solved, in the form of (7), is the following.

$$\begin{cases} -y_1 + y_3 - y_4 + y_5 - y_7 &= 0 \\ -y_2 + y_3 + y_4 + y_6 - y_8 &= 0 \\ -y_1 - y_2 + y_3 + y_5 + y_6 &\leq -1 \\ y_1,\, y_2,\, y_3,\, y_4,\, y_5,\, y_6,\, y_7,\, y_8 &\geq 0 \\ y &\in \mathbb{R}^8 \end{cases}$$

Solving such system yields the vertex (1, 1, 1, 0, 0, 0, 0, 0). Therefore, R contains an inconsistency, and the set of conflicting rules is $\{r_1, r_2, r_3\}$.

More than one IIS can be contained in an infeasible system. Some of them can overlap, in the sense that they can share some inequalities, although they cannot be fully contained one in another. Formally, the collection of all IIS of a given infeasible system is a *clutter* (see e.g. [1]). However, from the practical point of view, we are interested in IIS composed by a small number of rules inequalities. Moreover, it may happen that not all of them are equally preferable for the composition of the IIS that we are selecting. Hence, a cost c_k for taking each of the $[l+2(n+m)]$ inequalities into our IIS can be assigned. Such costs c_k for the inequalities of (6) corresponds to costs for the variables of system (7). A cost $[l + 2(n + m)]$-vector c is therefore computed, and the solution of the following linear program produces now an IIS having the desired inequality composition.

$$\begin{cases} \min\ c^T y \\ \quad y^T \begin{bmatrix} -B \\ I \\ -I \end{bmatrix} &= \mathbf{0} \\ \quad y^T \begin{bmatrix} -b \\ \mathbf{1} \\ \mathbf{U} \\ \mathbf{0} \end{bmatrix} &\leq -1 \\ y \geq \mathbf{0}, \quad y \in \mathbb{R}^{[l+2(n+m)]} \end{cases} \tag{8}$$

The result of Theorem 3.2 is not completely analogous to the linear case. In order to obtain more analogy, let us define the following property.

Integral-point property. A class of polyhedra which, if non-empty, contain at least one integral point, has the integral-point (IP) property.

Theorem 3.5. *If the polyhedron (6), which is the linear relaxation of (2), has the integral-point property, the following holds. If (7) is infeasible, (2) is feasible. On the contrary, if (7) is feasible, (2) is infeasible and each integer IIS is given by the restriction of the support of each vertex of polyhedron (7) to rules inequalities.*

Proof: If (7) is infeasible, (6) is feasible by Theorem 3.2. Since we assumed that the IP-property holds for (6), it contains at least one integral point. Since

the box constraints hold for (6), this integer point must be such that $x \in \{0,1\}^n, z \in \{0,...,U\}^m$, hence (2) is feasible. On the contrary, if (7) is feasible, the restriction of the support of a vertex in (7) to rule inequalities, that is a set of inequalities denoted by RI_1, has no integer solutions by Theorem 3.4. We now prove by contradiction that RI_1 is minimally infeasible, hence it is an integer IIS. Suppose RI_1 not minimal; then there exists a smaller set RI_1' such that $RI_1' \cup IC_1$ has no integer solutions. On the other hand, by Theorem 3.3, $RI_1' \cup BC_1$ is feasible, and since it has the IP-property, it has an integer solution, which is the contradiction. The thesis follows.

So far, when the IP property holds, solving a linear programming problem solves our inconsistency selection problem. There are several cases in which the linear relaxation (6) defines a polyhedron having the integral-point property (see e.g. [26]). Note that, imposing some syntactic restrictions, rules could be written in order to obtain one of such cases (e.g. logical rules can be *extended Horn* [8]).

When a rule is logically implied by other rules, such rule is said to be *redundant*. It is preferable to remove redundant rules, because decreasing the number of rules while maintaining the same power of error detection can speed up the whole process and make it less error prone.

Example 3.4. A very simple redundancy, with rules' meaning: (a) head of the house must have an annual income greater than or equal to 100, (b) everybody must have an annual income greater than or equal to 100. (a) is clearly redundant.

$$\begin{cases} (1 - x_{\texttt{role = head of the house}}) + x_{\texttt{annual income}\geq\texttt{100}} \geq 1 & \text{(a)} \\ x_{\texttt{annual income}\geq\texttt{100}} \geq 1 & \text{(b)} \end{cases}$$

Lemma 3.6. *A rule is redundant if and only if its inequality representation is implied by the inequality representation of all other rules.*

Due to the above easy lemma, redundancy can be detected with a standard procedure to detect implied inequalities in a linear system. Given a feasible linear system S and a single linear inequality $s^{\geq}$, $s^{\geq}$ is implied (i.e. the polyhedron described by S and the polyhedron described by $S \setminus s^{\geq}$ are the same) if and only if adding its negation $s^{<}$ (opportunely modified if necessary) to S produces an infeasible system. See also [4] for details. Redundancy of every rule is checked by iterating the above operation.

4 Correction of the Data Records

After the phase of *rules validation*, were the system (2) is checked to be feasible and to have more than one solution, detection of *erroneous records* p^e trivially becomes the problem of testing if the variable assignment that each record p determines for x and z satisfies the system (2). This operation can be performed

with an extremely small computational effort, hence it is suitable to check even a very large number of records. When an erroneous record p^e is detected, the *correction* process consists in changing some of its values, obtaining a *corrected record* p^c which satisfies the system (2) and is as close as possible to the (unknown) *original record* p^o, that is the one that would be present in absence of errors. In order to reach this purpose, two general principles should be followed during the imputation process: to apply the minimum changes to erroneous data, and to modify as less as possible the original frequency distribution of the data [13]. A cost for each change that we introduce in p^e is given, based on the reliability of each value. It is assumed that, when error is something unintentional and less probable than the correct value, the erroneous information is the minimum-cost set of values that, if changed, allows to satisfy system (2). Each record p^e corresponds to a variable assignment. In particular, we have a set of n binary values e_{ij} for the x_{ij} and a set of m integer values g_i for the z_i. We therefore have a cost $\hat{c}_{ij} \in \mathbb{R}_+$ for changing each e_{ij}, and a cost $\check{c}_i \in \mathbb{R}_+$ for changing each g_i.

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

A *donor record* p^d is a correct record which should be as similar as possible to p^o. This is obtained by selecting p^d being as close as possible to p^e, according to a suitable function δ giving a value v called the *distance* between p^e and p^d.

$$\begin{array}{rccc} \delta: & (D_1 \times \ldots \times D_m) \times (D_1 \times \ldots \times D_m) & \rightarrow & \mathbb{R}_+ \\ & (p^e, p^d) & \mapsto & v \end{array}$$

Also p^d corresponds to a variable assignment. In particular, we have a set of n binary values d_{ij} for the x_{ij} and a set of m integer values f_i for the z_i. The problem of *imputation through a donor* is to find a set K of fields of minimum total cost such that p^c can be obtained from p^e by copying from the donor p^d (only and all) the values of K. This is generally recognized to cause low alteration of the original frequency distributions, although changes caused to erroneous data may be not minimum. The correction by means of a donor is also referred to as *data driven* approach. We are interested in solving both of the above problems, and in choosing for each record the solution having the best quality.

Note that, generally, not all values involved in failed rules need to be changed, and that there are several alternative sets of values that, if changed, can make p^c such as to satisfy $\mathcal{F}$.

Example 4.1. Suppose that the following record p^e is declared erroneous using the two following rules: *i*) it is impossible that anybody not having a car lives in city `A` and works in city `B`; *ii*) the minimum `age` for driving is `18`.

{... `age` = `17`, `car` = `no`, `city of residence` = `A`, `city of work` = `B` ...}

Values involved in failed rules are here those of fields {`car`, `city of residence`, `city of work`}. Nevertheless, the record could be corrected either by changing values of the set {`city of residence`}, or by changing those of the set {`city of work`}, or by changing those of the set {`age`, `car`}. By supposing that all these fields have the same cost, the solution of error localization may be in this case $H_1 = \{$`city of residence`$\}$ or $H_2 = \{$`city of work`$\}$. However, suppose that the best donor available is the following p^d.

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

The solution of imputation through a donor would in this case be the set $K=$ {`age`, `car`}, having higher cost than H_1 and H_2.

In order to model the two above problems, let us introduce n binary variables $y_{ij} \in \{0, 1\}$ representing the changes we need to introduce in e_{ij}.

$$y_{ij} = \begin{cases} 1 & \text{if we change } e_{ij} \\ 0 & \text{if we keep } e_{ij} \end{cases}$$

Furthermore, only in the case of imputation through a donor, let us introduce m binary variables $w_i \in \{0, 1\}$ for the changes we need to introduce in g_i.

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

We now compose two integer linear programming models, whose optimal solution gives the solution to the two introduced variants of error correction problems. Denote by $\hat{c}$, $\check{c}$, y, w the vectors respectively made of all the components $\hat{c}_{ij}$, $\check{c}_i$, y_{ij}, w_i, for all $i = 1, \ldots, m$, $j = 1, \ldots, n_i$, as follows.

$$\begin{array}{ll} \hat{c} = (\hat{c}_{11}, \ldots, \hat{c}_{1n_1}, \ldots, \hat{c}_{m1}, \ldots, \hat{c}_{mn_m})^T & \check{c} = (\check{c}_1, \ldots, \check{c}_m)^T \\ y = (y_{11}, \ldots, y_{1n_1}, \ldots, y_{m1}, \ldots, y_{mn_m})^T & w = (w_1, \ldots, w_m)^T \end{array}$$

The minimization of the total cost of the changes can be expressed with the following objective functions. Objective (9) holds for the case of error localization, while objective (10) holds for the case of imputation through a donor.

$$\min_{y \in \{0,1\}^n} \quad \hat{c}^T y \tag{9}$$

$$\min_{\substack{y \in \{0,1\}^n \\ w \in \{0,1\}^m}} \quad \hat{c}^T y \; + \; \check{c}^T w \tag{10}$$

However, the constraints (2) are expressed by means of variables x and z. A key issue is that there is a relation between variables in (2) and variables in (9) and (10). In the case of error localization, this depends on the values of e_{ij}, as follows:

$$y_{ij} = \begin{cases} x_{ij} & \text{if } \; e_{ij} = 0 \\ 1 - x_{ij} & \text{if } \; e_{ij} = 1 \end{cases}$$

In fact, when $e_{ij} = 0$, to keep it unchanged means to put $x_{ij} = 0$. Since we do not change, $y_{ij} = 0$. On the contrary, to change it means to put $x_{ij} = 1$. Since we change, $y_{ij} = 1$. Altogether, $y_{ij} = x_{ij}$. When, instead, $e_{ij} = 1$, to keep it unchanged means to put $x_{ij} = 1$. Since we do not change, $y_{ij} = 0$. On the contrary, to change it means to put $x_{ij} = 0$. Since we change, $y_{ij} = 1$. Altogether, $y_{ij} = 1 - x_{ij}$. By using the above results, we can rewrite the objective function (9) by splitting the case of $e_{ij} = 0$ from the case of $e_{ij} = 1$. Denote with $E \in \{0,1\}^{n \times n}$ the following diagonal matrix.

$$E = diag\{e_{11}, \ldots, e_{1n_1}, \ldots, e_{m1}, \ldots, e_{mn_m}\}$$

The problem of error localization can now be modeled as follows,

$$\begin{cases} \min\ ((I-E)\hat{c})^T x + (E\hat{c})^T(\mathbf{1} - x) \\ \\ B \begin{bmatrix} x \\ z \end{bmatrix} \geq b \\ \\ \begin{array}{rcl} x & \in & \{0,1\}^n \\ z & \in & \{0,\ldots,U\}^m \end{array} \end{cases} \tag{11}$$

Conversely, in the case of imputation through a donor, relation between x_{ij} and y_{ij} depends on the values of e_{ij} and d_{ij}, as follows.

$$y_{ij} = \begin{cases} x_{ij} & \text{if } e_{ij} = 0 \text{ and } d_{ij} = 1 \\ 1 - x_{ij} & \text{if } e_{ij} = 1 \text{ and } d_{ij} = 0 \\ 0 & \text{if } e_{ij} = d_{ij} \end{cases}$$

In fact, when $e_{ij} = 0$ and $d_{ij} = 1$, not to copy the element means to put $x_{ij} = 0$. Since we do not change, $y_{ij} = 0$. On the contrary, to copy the element means to put $x_{ij} = 1$. Since we change, $y_{ij} = 1$. Altogether, $y_{ij} = x_{ij}$. When, instead, $e_{ij} = 1$ and $d_{ij} = 0$, not to copy the element means to put $x_{ij} = 1$. Since we do not change, $y_{ij} = 0$. On the contrary, to copy the element means to put $x_{ij} = 0$. Since we change, $y_{ij} = 1$. Altogether, $y_{ij} = 1 - x_{ij}$. Finally, when $e_{ij} = d_{ij}$, we cannot change e_{ij}, hence $y_{ij} = 0$. By using the above results, we can rewrite the objective function (10) by splitting the case of $e_{ij} = 0$ and $d_{ij} = 1$ from the case of $e_{ij} = 1$ and $d_{ij} = 0$.

Note, however, that even when x_{ij} does not change from e_{ij} to d_{ij}, z_i could still need to change from g_i to f_i in order to reach a feasible solution. For instance, this could help in satisfying mathematical rules. The choice of values for z_i affects the value of w_i, as follows.

$$z_i = g_i(1 - w_i) + f_i w_i$$

For simplicity's sake, we do not substitute the w variables in (10), but add to our model the above constraints linking the w to the z variables. Denote now with g and f the m-vectors respectively made with all the components g_i and f_i, for $i = 1 \ldots m$. Denote also with $D \in \{0,1\}^{n \times n}$ the following diagonal matrix,

$$D = diag\{d_{11}, \ldots, d_{1n_1}, \ldots, d_{m1}, \ldots, d_{mn_m}\}$$

The problem of imputation through a donor can now be modeled as follows.

$$\begin{cases} \min\ ((I-E)D\hat{c})^T x + (E(I-D)\hat{c})^T(\mathbf{1}-x) + \check{c}^T w \\ B \begin{bmatrix} x \\ z \end{bmatrix} \geq b \\ z = g^T(1-w) + f^T w \\ x \ \in \ \{0,1\}^n \\ z \ \in \ \{0,\ldots,U\}^m \end{cases} \tag{12}$$

5 Implementation and Computational Results

The procedure have been particularized to the case of a Census of population, where each record q is obtained through a compiled *questionnaire*, and describes a *household*, that is a set of *individuals*, as reported in [6]. The whole methodology was implemented in C++, using ILOG Concert Technology [20] in order to express the above illustrated optimization models. The overall software system developed for the census application, called DIESIS (Data Imputation Editing System - Italian Software) is currently used for correcting data from the Italian Census of Population 2001. A Census correction is known to be a very relevant and difficult proving ground for a data correction procedure [33]. Extensive tests have therefore been performed, and we report here only the most representative ones. Data used for the reported experimentation arise from the Italian Census of Population 1991. Three large data sets representing correct questionnaires were initially perturbed by introducing errors. They consist in 60,455 two-person households, 45,716 four-person households, and 20,306 six-person households. Data perturbation consists in randomly introducing non responses or other valid responses. Each data set was perturbed at four different increasing error levels, called 50, 100, 150, 200. Twelve different data sets are therefore obtained (2_050, 2_100, 2_150, 2_200, 4_050, 4_100, 4_150, 4_200, 6_050, 6_100, 6_150, 6_200). The set of rules used for experimentation are real rules, written by the Census experts of the Italian Statistic Office according to the specific syntax developed.

An initial routine reads the list of fields and rules to deal with, and automatically converts the rules into linear inequalities, thus generating the linear system (2). Such system is then checked for complete and partial inconsistencies. A sequence of the above shown matrix A and vector a, as in (7), are therefore generated from (2), those for the partial inconsistencies checking by adding to (2) the opportune $x_{ij} = 1$. By adding a suitable objective function, each problem in the form (8) is then solved by means of a state-of-the-art implementation of the simplex algorithm (ILOG Cplex [21]). When a vertex is found, its support is used to produce the IIS corresponding to the set of conflicting rules which is given in output. When the problem in form (8) is infeasible, either we check if it is a case were the IP-property holds (such check depending on the problem), and in such case no IIS exist, or we need to solve the integer feasibility

problem for the system (2), by means of a state-of-the-art implementation of a branch-and-cut procedure (again ILOG Cplex). However, for all considered real problems, when (8) is infeasible, (2) turn out to be feasible. Table 1 reports number of variables (# of var) and number of constraints (# of const) both for the system of linear inequalities being checked, and for the selected IIS, in addition to computational times (in seconds) on a Pentium IV 1.7GHz PC. We report only the (more interesting) cases corresponding to partial inconsistencies, hence to systems of type (2) containing an integer IIS. Those results are intended to give an example of application, rather than exploring all the computational possibilities of the proposed procedure.

Original system			Selected IIS		
Problem	# of var	# of const	# of var	# of const	Time (sec.)
EditPartIncons0	3,000	15,003	2	3	1.0
EditPartIncons1	3,000	15,002	21	22	4.1
EditPartIncons2	3,000	15,003	4	5	8.3
EditPartIncons3	3,000	15,003	3	4	6.5
EditPartIncons4	3,000	15,015	30	31	4.2
EditPartIncons5	3,000	15,002	1	2	0.9
EditPartIncons6	3,000	15,005	16	17	4.1
EditPartIncons7	3,000	15,003	6	7	6.5
EditPartIncons8	3,000	14,998	2	3	3.2
EditPartIncons9	3,000	15,003	2	3	1.6

Table 1: Cases of partial inconsistencies in the set of rules.

After inconsistencies repair, the system (2) is checked for redundancies by inverting the sense of each inequality, one at a time, and checking if the obtained system becomes infeasible. A sequence of integer feasibility problems is therefore solved, again by means of a branch-and-cut procedure (ILOG Cplex). The whole procedure, according to human experts having the charge of writing the rules, turn out to be a very satisfactory tool for the design of a contradiction-free and non-redundant set of rules.

Subsequently, the procedure detects erroneous records q^e by trivially checking if the variable assignment corresponding to each record (questionnaire) q satisfies the above validated system of linear inequalities (2) representing the rules. So far, the said datasets are each divided into a *correct set* and an *erroneous set*.

As for the phase of error correction, the practical behavior of the proposed procedure is evaluated both from the computational and from the data quality points of view, as follows. For each erroneous questionnaire record q^e, the error localization problem (11) is solved at first. After this, a number σ of donor questionnaires $\{q_1^d, \ldots, q_\sigma^d\}$ for each q^e are selected in the above mentioned correct set, by choosing the nearest ones according to our distance function δ. Therefore, σ problems of imputation through a donor (12) are solved. For each q^e, the number σ of used donors increases when the imputation quality (given by the value of the cost function (10) compared to the one of the error localization (9)) is not satisfactory. Hence, for each erroneous questionnaire q^e, a number $\sigma + 1$ of integer linear programming problems are solved, again by means of

a branch-and-cut procedure (ILOG Cplex). The corrected questionnaire q^c is finally obtained by choosing the solution corresponding to the best value of our objective function (10) among the σ donors. Computational times in minutes for solving each entire data set on a Pentium III 800MHz PC are reported in Table 2. Number of households in each dataset (# of households) and total number of integer linear programming problems solved for each dataset (# of problems solved) are also reported. As observable, each single correction problem is solved in extremely short time. Therefore, large data sets are corrected in very reasonable times.

Data set	# of households	# of problems solved	Total time (min.)
2_050	60,455	420,613	34.1
2_100	60,455	448,879	60.4
2_150	60,455	484,680	85.8
2_200	60,455	532,490	102.0
4_050	45,716	320,656	53.0
4_100	45,716	346,223	96.4
4_150	45,716	385,680	130.5
4_200	45,716	416,074	157.9
6_050	20,306	145,322	85.8
6_100	20,306	160,371	139.8
6_150	20,306	186,434	174.5
6_200	20,306	198,121	202.6

Table 2: Correction times for 2 persons, 4 persons and 6 persons households.

The statistical performances of the proposed methodology has also been strictly evaluated and compared with the performance of the Canadian Nearest-neighbor Imputation Methodology (CANCEIS) [3] by a simulation study [24] also based on real data from the 1991 Italian Population Census. CANCEIS has been selected for the comparative statistical evaluation because nowadays it is deemed to be the best specific methodology for automatically handling hierarchical demographic data. Results of such comparison, in [24], are very encouraging: the quality of the imputation performed by the proposed procedure is generally comparable, and sometimes better, than CANCEIS. The quality of imputed data was evaluated by comparing the original questionnaires (here known) with the corrected ones. We report in Table 3 the value of some particularly meaningful statistical indicators:

- the percentage of not modified values erroneously imputed by the procedure (E_{true});
- the percentage of modified values not recognized and therefore not imputed by the procedure (E_{mod});
- the percentage of cases when the value imputed by the procedure does not match the original value here known (I_{imp}).

Therefore, lower values correspond to a better data quality. Reported value is computed as average on the demographic fields `relation to head of the house, sex, marital status, age, years married`.

Data set	E_{true}	E_{mod}	I_{imp}
2_050	0.06	21.30	15.31
2_100	0.09	22.01	16.34
2_150	0.11	22.78	16.90
2_200	0.26	22.12	17.42
4_050	0.04	24.61	15.02
4_100	0.09	26.02	15.48
4_150	0.13	26.32	16.20
4_200	0.20	27.25	17.10
6_050	0.08	31.20	20.47
6_100	0.16	31.44	20.29
6_150	0.25	32.83	21.45
6_200	0.35	33.01	21.88

Table 3: Evaluation of the statistical quality of the correction.

The procedure introduces surprisingly few changes in fields that were not perturbed, is able to discover more than two times out of three the values which were modified, and imputes values which are generally correct. Note that, when randomly modifying values, the record can still appear correct, in the sense that it still satisfies the rules, so detection of perturbed values inherently has no possibility of being always exact. Note, moreover, that for fields having many values, as the case of `age`, the correct imputation is extremely difficult. Therefore, the imputation quality turns out to be of very high level.

6 Conclusions

Data correction problems are of great relevance in almost every field were an automatic data processing is used. A typical application consists in cleaning databases which can contain errors. Data correction problems have been tackled in several different manners, but satisfactory data quality and computational efficiency appear to be at odds. A new discrete mathematics modelization of the whole correction process allows the implementation of an automatic procedure for data correction. A linear inequalities representation of the set of rules makes the difficult problem of the logical validation of such set a treatable task. Detection of erroneous records can be performed with an inexpensive procedure. After this, the proposed procedure repairs the data using donors, ensuring so that the marginal and joint distribution within the data are, as far as it is possible, preserved. The sequence of arisen mathematical programming problems can be solved to optimality by using state-of-the-art MIP solving procedures. Computational limits of a data correction process are therefore pushed much further. Each single data record can be corrected in an extremely short time (always less than 1 second for the case of Census records). The statistical performances of the proposed general procedure has been strictly evaluated and compared with the performance of the Canadian Nearest-neighbor Imputation Methodology, which is nowadays deemed to be the best specific methodology for automatically handling hierarchical demographic data. Test results are very encouraging.

Acknowledgments The author wishes to thank Dr. S. Anselmi and Dr. M. Casaroli for helping the implementation work, and Dr. A. Reale and Dr. R. Torelli for providing edit rules.

References

[1] E. Amaldi, M.E. Pfetsch, L. Trotter, Jr. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. To appear in *Mathematical Programming.*

[2] M. Ayel and J.P. Laurent (eds.). *Validation, Verification and Testing of Knowledge-Based Systems.* John Wiley & Sons Ltd., Chichester, England, 1991.

[3] M. Bankier. Canadian Census Minimum change Donor imputation methodology. In *Proceedings of the Workshop on Data Editing*, UN/ECE, Cardiff, UK, 2000.

[4] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.

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

[6] R. Bruni, A. Reale, R. Torelli. Optimization Techniques for Edit Validation and Data Imputation. In *Proceedings of Statistics Canada Symposium: Achieving Data Quality in a Statistical Agency*, Ottawa, Canada, 2001.

[7] R. Bruni and A. Sassano. Error Detection and Correction in Large Scale Data Collecting. In *Advances in Intelligent Data Analysis*, Lecture Notes in Computer Science 2189, 84-94, Springer, 2001.

[8] V. Chandru and J.N. Hooker. *Optimization Methods for Logical Inference.* Wiley, New York, 1999.

[9] J.W. Chinneck. Fast Heuristics for the Maximum Feasible Subsystem Problem. *INFORMS Journal on Computing* 13(3), 210-223, 2001.

[10] J.W. Chinneck and E.W. Dravnieks. Locating Minimal Infeasible Constraint Sets in Linear Programs. *ORSA Journal on Computing* 3, 157-168, 1991.

[11] T. De Waal. *Processing of Erroneous and Unsafe Data.* Ph.D. Thesis, ERIM PhD series in Research Management, 2003.

[12] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.) *Advances in Knowledge Discovery and Data Mining.* AAAI Press / The MIT Press, Menlo Park, CA, 1996.

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

[14] E. Franconi, A. Laureti Palma, N. Leone, S. Perri, F. Scarcello. Census Data Repair: a challenging application of Disjunctive Logic Programming. In *Proceedings of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, (LPAR-2001) Lecture Notes in Artificial Intelligence 2250, Springer, 2001.

[15] R.S. Garfinkel, A.S. Kunnathur, G.E. Liepins. Optimal Imputation of Erroneous Data: Categorical Data, General Edits. *Operations Research* 34, 744-751, 1986.

[16] R.S. Garfinkel, A.S. Kunnathur, G.E. Liepins. Error Localization for Erroneous Data: Continuous Data, Linear Constraints. *SIAM Journal on Scientific and Statistical Computing* 9, 922-931, 1988.

[17] J. Gleeson and J. Ryan. Identifying Minimally Infeasible Subsystems of Inequalities. *ORSA Journal on Computing* 2(1), 61-63, 1990.

[18] O. Guieu and J.W. Chinneck. Analyzing Infeasible Mixed-Integer and Integer Linear Programs *INFORMS Journal on Computing* 11(1), 1999.

[19] I. Guyon, N. Matic, V. Vapnik. Discovering Informative Patterns and Data Cleaning. In [12].

[20] ILOG Concert Technology 1.0. *Reference Manual*. ILOG, 2000.

[21] ILOG Cplex 7.0. *Reference Manual*. ILOG, 2000.

[22] D.J. Hand, H. Mannila, P. Smyth. *Principles of Data Mining*. MIT Press, London, 2001.

[23] T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, Berlin, Heidelberg, 2002.

[24] A. Manzari and A. Reale. Towards a new system for edit and imputation of the 2001 Italian Population Census data: a comparison with the Canadian Nearest-neighbour Imputation Methodology. in *Proc. of the 53rd Session of the International Statistical Institute*, Seoul, South Korea, 2001.

[25] T. Menzies. Knowledge Maintenance: The State of the Art. *Knowledge Engineering Review*, 14(1), 1-46, 1999.

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

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

[28] C.T. Ragsdale P.G. McKeown. On Solving the Continuous Data Editing Problem. *Computers and Operations Research* 23, 263-273, 1996.

[29] R. Ramakrishnan and J. Gehrke. *Database Management System*. McGraw Hill, 2000.

[30] N. Rescher and R. Brandom. *The Logic of Inconsistency.* Basil Blackwell, Oxford, 1980.

[31] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, New York, 1986.

[32] M. Tamiz, S.J. Mardle, and D.F. Jones. Detecting IIS in Infeasible Linear Programs using Techniques from Goal Programming. *Computers and Operations Research* 23, 113-191, 1996.

[33] United Nations Economic Commission for Europe and the Statistical Office of the European Communities. Recommendations for the 2000 censuses of population and housing in the ECE region. *Technical Report Statistical Standards and Studies* No. 49, UN/ECE Statistical Division, 1998.

[34] H.P. Williams. *Model Building in Mathematical Programming*. J. Wiley, Chichester, 1993.

[35] 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, 1999.