Solving Error Correction for Large Data Sets by means of a SAT solver

Renato Bruni

Università di Roma "La Sapienza", Dip. di Informatica e Sistemistica,
Via Michelangelo Buonarroti 12 - 00185 Roma, Italy,
E-mail: `bruni@dis.uniroma1.it`

Abstract. The paper is concerned with the problem of automatic detection and correction of erroneous data into large datasets. The adopted process should therefore be computationally efficient. As usual, errors are here defined by using a rule-based approach: all and only the data records respecting a set of rules are declared correct. Erroneous records should afterwards be corrected, by using as much as possible the correct information contained in them. By encoding such problem into propositional logic, for each erroneous record we have a propositional logic formula, for which we want a model having particular properties. Correction problems can therefore be quickly solved by means of a customized SAT solver. Techniques for an efficient encoding of difficult cases are presented.

1 Introduction

Despite (or, perhaps, due to) the easiness with which data are nowadays available, their correctness often is a very relevant problem. Erroneous data should in fact be *detected* and possibly *corrected* by means of automated techniques. When dealing with large amount of information, moreover, not only the correctness of the correction procedure is needed, but also its computational efficiency.

As customary for structured information, data are organized into *records*. The problem of error detection is generally approached by formulating a set of *rules* that every record must respect in order to be declared *correct*. Records not respecting all the rules are declared *erroneous*. Such rule-based approach has several advantages (e.g. flexibility, intelligibility) on other methods. In the field of database theory, rules are also called *integrity constraints* [18], whereas in the field of statistics, rules are also called *edits* [8], and express the error condition. The study of such rules is a central problem for the areas of data mining and data analysis [4, 13, 14]. 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, according to the two following criteria: a) the corrected record should be as *similar* as possible to the erroneous record; b) the correction operation should respect the *frequency distributions* of the data. 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). In the field of statistics, the correction process is also called *data* *imputation*. Several different rules encoding and solution algorithm have been proposed (e.g. [3, 17]). 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 [8]. In practical case, however, such methods suffer from severe computational limitations [17]. In the field of computer science, the correction process is also called *data cleaning*. Errors may be detected as inconsistencies in knowledge representation, and corrected with consistency restoring techniques [1, 15, 19]. Another approach to error correction, in database theory, consists in performing a cross validation among multiple sources of the same information [18]. A previous work [5] describes the case when rules are encoded into *linear inequalities*. The correction is therefore achieved by solving MILP problems by means of a *branch-and-cut* procedure (ILOG Cplex). In another work [6] error localization and error correction problems are modeled as *set covering* problems and solved by means of the *volume algorithm* compared to *branch-and-bound*.

An automatic procedure for performing generic data correction by using only *propositional logic* is here presented. The above described problems are therefore solved by solving a variant of *propositional satisfiability* (SAT) problems. (see e.g. [7, 10, 12, 22] for extensive references). SAT solvers available today are the result of decades of research work and are deemed to be among the fastest NP-complete problem-specific solvers (see for instance computational results in [20, 21]). For this reason, the computational issue appears to be tackled in a satisfactory manner.

A formalization of data and rules is given in Sect. 2. In the same section is also sketched the basic procedure for propositional logic encoding of rules, already outlined in [6]. There are, however, cases when the basic procedure suffers from an unpleasant growth of the number of clauses and variables. We therefore analyze, in Sect. 3, such cases, and present new techniques to overcome the above problems. After this, the correction problem is formalized, and a complexity analysis is given, in Sect. 4. An efficient new correction procedure, implementable by introducing minor modifications into a generic DPLL SAT solver, is then reported. Such modifications are actually introduced in the recent SAT solver named SIMO (Satisfiability Internal Module Object oriented) [11]. Computational results on datasets of two different origins (*population* data and *television transmitters* data) are reported in Sect. 5.

2 General Propositional Logic Encoding

In Database theory, a *record schema* is a finite set of fields f_i, with $i = 1 \dots l$. A *record instance* is a set of values v_i, one for each of the above fields [18]. In order to help exposition, we will focus on records representing *persons*, which are somehow more intuitively understandable. Note, however, that the proposed procedure is completely general, being not influenced by the meaning of treated data. The record scheme will be denoted by P, whereas a corresponding record instance will be denoted by p.

$$P = \{f_1, \dots, f_l\} \qquad p = \{v_1, \dots, v_l\}$$

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

Each field f_i has a *domain* D_i, with $i = 1 \ldots l$, which is the set of possible values for that field. Since we are dealing with errors, the domains include all values that can be found in data, even the erroneous ones. Fields are usually 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 on its domain. Examples of quantitative fields are numbers (real or integer), or even the elements of an ordered set. A qualitative field simply requires its domain to be a discrete set with finite number of elements. The case of fields having a not-finite number of not-ordered values is generally ignored. 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 values, 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 p is declared correct if and only if it respects a *set of rules* denoted by $R = \{r_1, \ldots, r_t\}$. Each rule can be seen as a mathematical function r_h from the Cartesian product of all the domains to the Boolean set {0,1}.

$$r_h : D_1 \times \ldots \times D_l \rightarrow \{0, 1\}$$

Rules are such that p is a correct record if and only if $r_h(p) = 1$ for all $h = 1 \ldots t$.

Example 2.3. An error within a person record can be the following one:

$$\texttt{marital status} = \texttt{married} \text{ and } \texttt{age} = \texttt{10} \text{ years old}$$

The rule to detect this kind of errors could be: if `marital status` is `married`, `age` must be not less than, say, `14`.

Rules can be obtained from various sources (e.g. human expertise, machine learning, see also [4, 13, 14]). We consider the set of rules already given, and intuitively expressed as logical connections (representable with $\wedge$, $\vee$, $\Rightarrow$, etc.) among *statements*. Each statement involves a single field f_i, and consist in a set of values from its domain D_i.

Example 2.4. A statement can be (`marital status` = `married`). Another statement can be (`age` $\geq$ `14`).

We now describe the basic procedure for encoding rules into clauses. A *clause* C_u is a disjunction of possibly negated propositional variables x_v. We assume the reader familiar with propositional logic. By denoting with π_u the set of indices of the positive variables of C_u, and with ν_u that one of negative variables of C_u, the clause is:

$$\bigvee_{v\in\pi_u} x_v \vee \bigvee_{v\in\nu_u} \neg x_v$$

Values appearing in the rules are called *breakpoints*, or *cut points*, for the domains. All breakpoints concerning domain D_i represent logical *watershed* between the values of domain D_i. Their set will be denoted with B_i, as follows:

$$B_i = \{b_i^1, \ldots, b_i^{k'_i}\}$$

The set B_i determines a partition of domain D_i into subsets S_i^s. By furthermore merging possibly equivalent subsets, we obtain the following collection of subsets:

$$\mathcal{S}_i = \{S_i^1, \ldots, S_i^{k_i}\}$$

All and only values belonging to the same subset are *equivalent* from the rules' point of view. We congruently have $D_i = (S_i^1 \cup \ldots \cup S_i^{k_i})$. A subset for the *out-of-range* values is always present, while other subsets form the *feasible* domain.

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.

$$\{b_{\texttt{age}}^1 = \texttt{0},\ b_{\texttt{age}}^2 = \texttt{14},\ b_{\texttt{age}}^3 = \texttt{18},\ b_{\texttt{age}}^4 = \texttt{26},\ b_{\texttt{age}}^5 = \texttt{110},\ b_{\texttt{age}}^6 = \texttt{blank}\}$$

By using $B_{\texttt{age}}$ and R, the subsets are obtained. $S_{\texttt{age}}^5$ is the out-of-range one.

$$S_{\texttt{age}}^1 = \{\texttt{0}, \ldots, \texttt{13}\},\ S_{\texttt{age}}^2 = \{\texttt{14}, \ldots, \texttt{17}\},\ S_{\texttt{age}}^3 = \{\texttt{18}, \ldots, \texttt{25}\}$$
$$S_{\texttt{age}}^4 = \{\texttt{26}, \ldots, \texttt{110}\},\ S_{\texttt{age}}^5 = \{\ldots, \texttt{-1}\} \cup \{\texttt{111}, \ldots\} \cup \{\texttt{blank}\}$$

So far, subsets can be encoded with propositional variables in several manners (for instance, k_i subsets can be encoded by $\lceil \log_2 k_i \rceil$ variables). We choose to encode the k_i subsets of domain D_i with $n_i = k_i - 1$ variables, with the aim of obtaining a more comprehensible encoding, as follows. When value v_i of field f_i belongs to subset S_i^s, it means $x_{is} = True$ and $x_{ih} = False$, for $h = 1, \ldots, n_i$, $h \neq s$. The same holds for all other subsets of f_i, except for the out-of-range subset, which is encoded by putting all variables x_{ih} at $False$, for $h = 1, \ldots, n_i$.

Example 2.6. The field `marital status` has 6 subsets, hence 6-1 = 5 variables $x_{\texttt{mar.st.=single}}$, $x_{\texttt{mar.st.=married}}$, $x_{\texttt{mar.st.=separate}}$, $x_{\texttt{mar.st.=divorced}}$, $x_{\texttt{mar.st.=widow}}$

We define $n' = n_1 + \ldots + n_l$, obtaining so n' propositional variables $\{x_{is}\}$. Now, every statement, (i.e. every set of values of a domain appearing in a rule) can be substituted by the corresponding propositional variable (or by the corresponding logic connection of propositional variables). Since each rule is itself a logic connection of statements, each rule can be expressed as a logic connection of propositional variables, which is put in the form of clauses.

Example 2.7. Consider the rule (`marital status` = `married`) ⇒ (`age` ≥ `14`). By substituting the introduced propositional variables, we have the logic formula $x_{\texttt{mar.st.=married}} \Rightarrow \neg x_{\texttt{age}\in\{\texttt{0,13}\}}$, easily coverted into the following clause.

$$\neg x_{\texttt{mar.st.=married}} \vee \neg x_{\texttt{age}\in\{\texttt{0,13}\}}$$

In addition to information given by rules, 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 each field f_i must have only one value, and therefore $\binom{n_i}{2}$ (number of unordered pairs from n_i possibilities) clauses, named *congruency* clauses, are added.

3 Special Cases for Encoding and Complete Procedure

The above described basic procedure is suitable for a large variety of situations. There are, however, cases where it suffers from an undesirable growth in the number of clauses and/or variables. We now analyze the most relevant ones.

3.1 Denominative Fields

A first problem arise when a qualitative field f_i has a feasible domain containing a *large* number of values. This happens for instance in the case of `name` of a person. One can theoretically state a rule for every correct name, and so detect errors like `Jhon` instead of `John`. This would, however, require the use of a large number of rules and, consequently, of clauses and variables. It is instead convenient, in such cases, to consider a list L_i of values which are *feasible* for f_i

$$L_i = \{v_i^1, \dots, v_i^{a_i}\}$$

Such list could be for instance obtained from other correct data sets of similar nature. So far, the (possibly erroneous) value v_i is *identified* as a value $v_i^b \in L_i$ and substituted with it. If v_i is so erroneous that cannot be identified as any feasible value, it is then identified as the `blank` value and substituted with it. The identification process involves *linkage* problems that cannot be discussed here. After such substitution, there is no need of rules for every correct name anymore. Note that other rules for the field f_i may still be written (for instance absence of `blank`). Such rules would just determine, as usual, a partition in the domain D_i.

3.2 Corresponding Fields

A second problem arise when two fields f_i, f_j having a *large* number of feasible values must be *congruent*. This happens for instance in the case of a `location` and its geographical `coordinates` (latitude and longitude). Writing the rules defining all the correct couples (`location, coordinates`) would produce a large number of clauses and variables. A tentative for recognizing such couple within a list would ignore a global correction of the record. It is instead convenient (after identifying denominative fields such as `location` by using a list) to produce the correct couple of values

$$\{v_i^i, v_j^i\}$$

which is obtainable on the basis of f_i, and the correct couple of values

$$\{v_i^j, v_j^j\}$$

which is obtainable on the basis of f_j. One of such couples must be chosen, and there is no need of rules defining which are the correct couples anymore. In order to allow such *choice* during the correction process, instead, further propositional variables need to be introduced. By pursuing the aim of a more comprehensible encoding (as in Sect. 2), the 2 choices are encoded here by 2 variables y_i and y_j. Clauses expressing their mutual exclusion are also added: $(y_i \vee y_j), (\neg y_i \vee \neg y_j)$.

Such y variables must be connected to the x variables, by expressing that $y_i \Rightarrow (v_i = v_i^i, v_j = v_j^i)$, and analogously for y_j, which means:

$$\begin{array}{llll} y_i \Rightarrow x_{is} & \text{for } s \text{ such that } v_i^i \in S_i^s & \quad y_j \Rightarrow x_{is} & \text{for } s \text{ such that } v_i^j \in S_i^s \\ y_i \Rightarrow x_{js} & \text{for } s \text{ such that } v_j^i \in S_j^s & \quad y_j \Rightarrow x_{js} & \text{for } s \text{ such that } v_j^j \in S_j^s \end{array}$$

Such logical expressions are trivially converted into clauses. Note that also for the corresponding fields f_i and f_j other rules may be written. The above can be easily generalized to the case of any number q of corresponding fields, by introducing the q variables $\{y_1, \ldots, y_q\}$ and the suitable clauses.

Example 3.1. Consider a record having `location` = `Rome` and `coordinates` = `40.5N74W`. By using a Geographic Information System (GIS) we have the couples

$$\begin{array}{cc} \texttt{location} = \texttt{Rome} & \texttt{coordinates} = \texttt{40.5N74W} \\ \downarrow & \downarrow \\ \left\{\begin{array}{l} \texttt{location}^{\texttt{loc}} = \texttt{Rome} \\ \texttt{coordinates}^{\texttt{loc}} = \texttt{41.5N12.5E} \end{array}\right. & \left\{\begin{array}{l} \texttt{location}^{\texttt{coor}} = \texttt{NewYork} \\ \texttt{coordinates}^{\texttt{coor}} = \texttt{40.5N74W} \end{array}\right. \end{array}$$

We introduce two new variables $y_{\texttt{loc}}$ and $y_{\texttt{coor}}$, respectively indicating whether the `location` or the `coordinates` are chosen as correct. Suppose that the field `location` has a subset for `Rome` and a subset for `NewYork`, and that the field `coordinates` has a subset for `40-50N10-20E` and a subset for `40-50N70-80W`. We therefore generate the following clauses:

$$\left\{\begin{array}{l} (y_{\texttt{loc}} \vee y_{\texttt{coor}}), (\neg y_{\texttt{loc}} \vee \neg y_{\texttt{coor}}), \\ (\neg y_{\texttt{loc}} \vee x_{\texttt{loc=Rome}}), (\neg y_{\texttt{loc}} \vee x_{\texttt{coor=40\_50N10\_20E}}), \\ (\neg y_{\texttt{coor}} \vee x_{\texttt{loc=NewYork}}), (\neg y_{\texttt{coor}} \vee x_{\texttt{coor=40\_50N70\_80W}}) \end{array}\right.$$

3.3 Contained Fields

A third problem arise in the case of a field f_i representing an object, and another field f_j representing a set containing many such objects. This happens for instance in the case of `city` and `state`. Writing the rules for all correct couples (`city`, `state`) would again produce a large number of clauses and variables. It is therefore convenient to proceed similarly to the former case, with the only difference that the values obtained on the basis of the containing field f_j will have a `blank` for the contained field f_i.

Example 3.2. For a record having $\texttt{city} = \texttt{Rome}$ and $\texttt{state} = \texttt{France}$, we have:

$$\begin{array}{cc}
\texttt{city} = \texttt{Rome} & \texttt{state} = \texttt{France} \\
\downarrow & \downarrow \\
\begin{cases} \texttt{city}^{\texttt{city}} = \texttt{Rome} \\ \texttt{state}^{\texttt{city}} = \texttt{Italy} \end{cases} & \begin{cases} \texttt{city}^{\texttt{state}} = \texttt{blank} \\ \texttt{state}^{\texttt{state}} = \texttt{France} \end{cases}
\end{array}$$

3.4 Mathematical Rules

There are, however, cases for which the logic encoding is not convenient. A typical example are rules containing mathematical operations between fields. In such case, a propositional logic encoding is possible, e.g. [23], but, due to the very large growth in the size of the problem, a mathematical encoding of the problem seems preferable. This clearly prevent using a SAT solver, and requires the use of a MILP procedure, as in [5].

3.5 Complete Encoding Procedure

We can now state the complete procedure for propositional encoding of rules.

1. *Identification of domain D_i for each field f_i, considering that we are dealing with errors*
2. *Generation of list L_i and identification of v_i for each denominative field f_i*
3. *Identification of k_i subsets $\{S_i^s\}$ in each domain D_i, by using breakpoints B_i and rules R, and by merging (possible) equivalent subsets within D_i*
4. *Definition of n_i variables $\{x_{is}\}$ to encode subsets $\{S_i^s\}$ of each domain D_i*
5. *Expression of each rule r_h by means of clauses defined over the introduced variables $\{x_{is}\}$*
6. *Definition of variables $\{y_j\}$ and clauses for each special field f_j*
7. *Generation of congruency clauses to supply information not present in rules*

By writing rules according to a precise syntax, the above encoding was performed by an automatic procedure. Altogether, the set of rules produces a set of m clauses over $n = n' + n''$ propositional variables $\{x_{is}, y_j\}$, hence a CNF formula $\mathcal{F}$. Each record p produces a truth assignment $\{x_{is} = t_{is}, y_j = t_j\}$ for such variables (t_{is} and t_j denoting a truth value in $\{$*True, False*$\}$). We say, briefly, that p must *satisfy* $\mathcal{F}$, i.e. make it *True*, to be correct. Erroneous record detection simply consists in checking the truth assignment given by each record on $\mathcal{F}$.

4 The Correction Procedure

Given an *erroneous record* $p^e = \{v_1^e, \ldots, v_l^e\}$, the *correction* process consists in changing some of its values, obtaining a *corrected record* $p^c = \{v_1^c, \ldots, v_l^c\}$ which satisfies the formula $\mathcal{F}$ and is as close as possible to the (unknown) *original record* p^o, that is the one we would have in absence of errors. 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`.

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

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`}.

Two principles should be followed during the correction process: to apply the minimum changes to erroneous data, and to modify as little as possible the original frequency distributions of values for each field (see e.g. [8]). The first aspect is generally settled by assigning a cost for changing each value v_i^e. Such cost is based on the reliability of the field f_i. It is assumed that, when error is something unintentional, the erroneous fields are the minimum-cost set of fields that, if changed, can satisfy $\mathcal{F}$. The second statistical aspect is currently settled by using, for each erroneous record p^e, a donor [3]. A *donor record* $p^d = \{v_1^d, \ldots, v_l^d\}$ is a correct record which, according to some (easily computable) distance function $\sigma(p^e, p^d) \in \mathbb{R}_+$, is the nearest one to the erroneous record p^e, hence it represents a record having similar characteristics. Such donor is used for providing values for the fields to be changed, according to the so-called *data driven* approach (see e.g. [3]).

We now turn back to propositional logic. In order to simplify notation, we will hereafter denote by p_e the truth assignment $\{x_{is} = t_{is}^e, y_j = t_j^e\}$ (where some t_j^e may be undefined) corresponding to the erroneous record p^e and similarly for p^d, p^c, etc. The record correction problem can be logically defined as follows.

Definition 4.2. *Given a logic formula $\mathcal{F}$ (encoding of the set of rules R), a truth assignment p_e not satisfying $\mathcal{F}$ (encoding of the erroneous record p^e), a truth assignment p_d satisfying $\mathcal{F}$ (encoding of the donor record p^d), the record correction problem consists is finding a truth assignment for the $\{x_{is}, y_j\}$ variables such that*

$$\min\{\delta(p_e, \{x_{is}, y_j\}) : \{x_{is}, y_j\} \text{ satisfies } \mathcal{F}\}$$

where $\delta(p_e, \{x_{is}, y_j\}) \in \mathbb{R}_+$ is a linear (or other easily computable) distance function measuring the cost of the changes introduced in p^e for obtaining p^c.

Theorem 4.3. *The above defined record correction problem is NP-hard.*

Proof: Given a propositional logic formula $\mathcal{F}$ and a model m_1, the problem of deciding whether it exists another model m_2 such that the Hamming distance between m_1 and m_2 is less than or equal to a given number k has been proved NP-complete in [2]. Therefore, given a satisfiable $\mathcal{F}$ and a generic truth assignment t_1, the problem of deciding whether it exists a model within a given Hamming distance from t_1 has the same complexity. If the adopted distance function is instead a generic easily (i.e. polynomially) computable function, the problem clearly remains NP-hard [9]. Since this latter problem constitutes the decision version of the above defined record correction problem, the thesis follows.

Moreover, the minimization version of the error correction problem has strong similarities with the combinatorial optimization problem[1].

The solution of the defined record correction problem is searched by means of a DPLL style procedure. Very basically, a DPLL SAT solver explores the solution space by generating a search tree, repeating the following steps until a satisfying solution is reached or the solution space is completely explored.

1. *Choose a variable x_s to be fixed*
2. *Fix x_s to a truth value t_s and propagate such fixing*
3. *If needed, backtrack and cut that branch*

The modifications introduced for solving our problem are basically the following:

a) Variables $\{x_{is}, y_j\}$ are initially progressively fixed to values $\{t^e_{is}, t^e_j\}$. After the first backtrack, truth values corresponding to subsets S^s_i in progressively increasing distance from those of p^e will be chosen, according to the minimum change principle.

b) The distance $\delta(p_e, \bar{p})$ of current (partial) solution $\bar{p}$ from p_e is computed at every node of the branching tree, on the basis of the truth assignments $\{t^e_{is}, t^e_j\}$ and $\{\bar{t}_{is}, \bar{t}_j\}$. The donor's truth assignment p_d is used as current optimal solution, hence backtrack is performed not only when an empty clause is obtained, but also when the distance $\delta(p_e, \bar{p})$ of current (partial) solution is such that

$$\delta(p_e, \bar{p}) \geq \sigma(p_e, p_d)$$

[1] Let $A = \{a_1, \ldots, a_f\}$ be a finite ground set, $\mathcal{D} = \{D_1, \ldots, D_g\}$ be a collection of subsets of A, given a linear cost function $w(D) \in \mathbb{R}$, the combinatorial optimization problem is $\{\min w(D) : D \in \mathcal{D}\}$ [16]. It is quite immediate to see the correspondence between the set p^e and the ground set A. By denoting with $\{D_1, \ldots, D_t\}$ the alternative sets of values of p^e that, if changed, can satisfy $\mathcal{F}$ (Such collection of sets is nonempty because by changing at most all fields we are able to satisfy $\mathcal{F}$, and finite because so it is the number of possible subsets of fields in p^e. Compare also to Example 4.1.) and by considering as $w(D)$ the function $\delta(p_e, p_c)$ measuring the cost of the values changed in p^e for satisfying $\mathcal{F}$, an instance of record correction problem becomes an instance of combinatorial optimization problem.

c) After reaching a (complete) solution $p^\star$ such that $\delta(p_e, p^\star) < \delta(p_e, p_d)$ (which means that $p^\star$ is more similar than p_d to p_e), the search does not stop, but continues using now $p^\star$ as current optimal solution, and updating $p^\star$ when possible.
d) The search stops either when a solution $p^\star$ having a distance $\delta(p_e, p^\star)$ being small *enough* is reached, or when the search tree is completely explored without finding a solution having acceptable distance.

Once the truth assignment corresponding to the optimal solution $p^\star$

$$\{x_{is} = t^\star_{is}, y_j = t^\star_j\}$$

is obtained, that corresponds to knowing, for each field f_i, which subset S^s_i the corrected value v^c_i must belong to. The actual value v^c_i will then be the initial one v^e_i if $v^e_i \in S^s_i$, otherwise the donor's one v^d_i if $v^d_i \in S^s_i$, or otherwise generated by using a data driven or a probabilistic approach. Depending on the field, in the latter cases, one may report that the information cannot be reconstructed, although known belonging to subset S^s_i.

5 Implementation and Results

The procedure is implemented by introducing minor modifications in in the recent SAT solver named SIMO 2.0 (Satisfiability Internal Module Object oriented) [11]. Such solver was chosen also because of its original purpose of providing an open-source library for SAT solvers. The proposed procedure has been applied to the correction of datasets arising from two different origins: *population* data and *television transmitters* data. The correction of such kind of data actually represents a very relevant problem for people working in the respective fields. As observable, each single record correction problem is quite small-sized for a modern SAT solver. However, very large datasets should often be corrected, and the whole process is therefore computationally demanding. Reported results are obtained on a Pentium IV 1.7GHz PC with 1Gb RAM. In both cases, results are very encouraging both from the computational and from the data quality point of view.

5.1 Population Data

The process of error correction in the case of simulated population data sets using 258 real rules, kindly provided by experts of the Italian National Statistic Institute (Istat), is performed. About 400 congruency clauses were also needed. Four different data sets (p_050, p_100, p_150, p_200) are obtained by randomly changing the values of 5 demographic variables (`relation to head of the house`, `sex`, `marital status`, `age`, `years married`) into 182,864 correct simulated person records. Such values are changed by introducing either blank values or other valid values with a base probability η respectively multiplied by 0.5, 1, 1.5, 2, in order to obtain the above data sets. The resulting erroneous record content of such four data sets after the described perturbation respectively became 22.5%, 40.5%, 54.0%, 64.4%. Since the original (correct) records were in these cases known, the accuracy of the correction process could be evaluated.

Results of the correction of each whole data set are reported in Table 1. Number of variables and number of clauses of the formula $\mathcal{F}$ considered when solving each single record correction problem differs from record to record. This because, depending on the errors, one could know in advance that some (few) fields are not interested by the correction process. We therefore generated variables only for those fields which could be interested by the correction process, and consequently only the clauses containing them. Error content of the datasets is generally enough for allowing the reconstruction, and the record correction process was successful in more than the 99.9% of the cases. There are, however, records for which no solution was found, because the best solution obtained has a distance higher than the maximum acceptable distance.

Number of records	182,864
Number of fields	20
Number of rules	258
Error content before the correction process in p_050	22.5%
Error content before the correction process in p_100	40.5%
Error content before the correction process in p_150	54.5%
Error content before the correction process in p_200	64.4%
Error content after the correction process (all)	< 0.01%
Total correction time for p_050	50.4 min.
Total correction time for p_100	79.3 min.
Total correction time for p_150	112.6 min.
Total correction time for p_200	113.8 min.
Average time per record for p_050	0.01 sec.
Average time per record for p_100	0.02 sec.
Average time per record for p_150	0.04 sec.
Average time per record for p_200	0.04 sec.
Average number of variables for each record correction	315
Average number of clauses for each record correction	633 (258+0+375)

Table 1: Results on the four datasets p_050, p_100, p_150, p_200 of simulated persons data using real rules.

5.2 Television Transmitters Data

The correction process for data representing Television Transmitters data is performed. The processed dataset has 22,504 records, each having 209 fields describing the geographical, orographical and radio-electrical characteristics of each transmitter. Differently from the former cases, this dataset already contained errors, and they have not been artificially introduced. A set of 75 explicit rules has been written by experts of Frequency Planning for Broadcasting. The resulting number of clauses is 224, 75 of which derive from the initial explicit rules, 99 from the logical connections between the x variables and the y variables, and 50 are congruency clauses. Results of the correction of the whole dataset are reported in Table 2. Proposed techniques for dealing with corresponding fields have been of crucial importance for such dataset. Error content of the data is higher than in the former datasets, and in some cases the information is not enough for allowing a reconstruction. In such cases, in fact, the best solution obtained has a distance higher than the maximum acceptable distance, and therefore a certain record percentage could not be corrected.

Number of records	22,504
Number of fields	209
Number of rules	75
Error content before the correction process	62.5%
Error content after the correction process	2.1%
Total correction time	4.0 min.
Average time per record	0.01 sec.
Average number of variables for each record correction	76
Average number of clauses for each record correction	224 (75+99+50)

Table 2: Results on the television transmitters dataset using real rules.

6 Conclusions

Data correction problems are of great relevance in almost every field were an automatic data processing is used. When cleaning large scale databases, moreover, computational efficiency is essential. A propositional logic encoding is, in many cases, the most direct and effective representation both for records and for the set of rules. Erroneous records detection is carried out with an inexpensive procedure, while erroneous records correction is solved by customizing a generic DPLL style SAT solver (SIMO 2.0 for the reported results). The record correction problem is in fact NP-hard. Additional techniques for handling difficult cases for encoding are presented. Computational results on datasets having different origins (*population* data and *television transmitters* data) are very encouraging.

Acknowledgments. The author whish to thank Daniele Praticò for his contribution to present paper, and SIMO developers for providing their solver.

References

1. M. Ayel and J.P. Laurent (eds.). *Validation, Verification and Testing of Knowledge-Based Systems.* J. Wiley & Sons, Chichester, 1991.
2. O. Bailleux and P. Marquis. DISTANCE-SAT: Complexity and Algorithms. In *Proceedings* AAAI/IAAI 1999, 642-647.
3. M. Bankier. Canadian Census Minimum change Donor imputation methodology. In *Proceedings of the Workshop on Data Editing*, UN/ECE, Cardiff, 2000.

4. E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79, 163-190, 1997.
5. R. Bruni, A. Reale, R. Torelli. Optimization Techniques for Edit Validation and Data Imputation. In *Proceedings of Statistics Canada International Symposium: Achieving Data Quality in a Statistical Agency*, Ottawa, Canada, 2001.
6. R. Bruni and A. Sassano. Error Detection and Correction in Large Scale Data Collecting. In *Advances in Intelligent Data Analysis*, LNCS 2189, Springer, 2001.
7. V. Chandru and J.N. Hooker. *Optimization Methods for Logical Inference.* Wiley, New York, 1999.
8. P. Fellegi and D. Holt. A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association* 71(353), 17-35, 1976.
9. M.R. Garey and D.S. Johnson. *Computers and Intractability.* Freeman, New York, 1979.
10. I.P. Gent, H. van Maaren, T. Walsh (eds.) *SAT 2000.* IOS Press, Amsterdam, 2000.
11. E. Giunchiglia, M. Maratea, A. Tacchella. Look-Ahead vs. Look-Back techniques in a modern SAT solver. In *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing* (SAT2003), Portofino, Italy, 2003.
12. J. Gu, P.W. Purdom, J. Franco, and B.W. Wah. Algorithms for the Satisfiability (SAT) Problem: A Survey. *DIMACS Series in Discrete Mathematics* American Mathematical Society, 1999.
13. D.J. Hand, H. Mannila, P. Smyth. *Principles of Data Mining.* MIT Press, London, 2001.
14. T. Hastie, R. Tibshirani, J. Friedman. *The Elements of Statistical Learning.* Springer, New York, Berlin, Heidelberg, 2002.
15. T. Menzies. Knowledge Maintenance: The State of the Art. *Knowledge Engineering Review*, 14(1), 1-46, 1999.
16. G.L. Nemhauser, L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, New York, 1988.
17. 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.
18. R. Ramakrishnan, J. Gehrke. *Database Management System.* McGraw Hill, 2000.
19. N. Rescher, R. Brandom. *The Logic of Inconsistency.* B. Blackwell, Oxford, 1980.
20. L. Simon and P. Chatalic. SAT-Ex: the experimentation web site around the satisfiability problem. `http://www.lri.fr/~simon/satex`.
21. L. Simon, D. Le Berre (and E.A. Hirsch for SAT2002). Sat competition web site `http://www.satlive.org/SATCompetition`
22. K. Truemper. *Effective Logic Computation.* Wiley, New York, 1998.
23. J. Warners. A Linear-time Transformation of Linear Inequalities into Conjunctive Normal Form. *Information Processing Letters*, 68, 63-69, 1998.