Fixing Inconsistent Databases by Updating Numerical Attributes

Leopoldo Bertossi, **Loreto Bravo**

Carleton University, School of Computer Science, Ottawa, Canada.
{bertossi,lbravo}@scs.carleton.ca

Enrico Franconi, **Andrei Lopatenko***

Free University of Bozen–Bolzano, Faculty of Computer Science, Italy.
{franconi,lopatenko}@inf.unibz.it

Abstract

Consistent query answering is the problem of computing those answers from a database that are consistent with respect to certain integrity constraints that the database as whole may fail to satisfy. Those answers are characterized as invariant under minimal forms of restoring the consistency of the database. In this context, we study the problem of fixing databases by updating numerical data at the attribute under denial constraints.

1. Introduction

Many database applications, like census, demographic, financial, and experimental data, contain quantitative data associated to nominal or qualitative data, e.g. number of children associated to a household identification code; or measurements associated to a sample identification code. It is common for this kind of numerical data to contain errors or mistakes with respect to certain semantic constraints. For example, a census form for a particular household may be considered incorrect if the number of children exceeds 20; or if the age of the wife is less than 10. This kind of restrictions can be expressed using denial integrity constraints (ICs), that prevent some attributes from taking certain values. In this kind of applications, inconsistencies in numerical data are resolved by changing individual attribute values, while values in the key attributes are kept, e.g. without changing the household code, the number of children is decreased considering the admissible values.

In this paper we consider the problem of fixing numerical data according to certain constraints while (a)

---

* Also: University of Manchester, Department of Computer Science, UK.

keeping the values associated to the keys of the relations in the database, and (b) minimizing the quantitative global distance from the modified instance to the original instance. Since the problem may admit several global solutions, each of them involving possibly many individual changes, we are particularly interested in characterizing and computing data and properties that remain invariant under any of these fixing processes. We concentrate on linear denial constraints and conjunctive queries.

Database repairs have been extensively studied in the context of consistent query answering (CQA), i.e. the process of obtaining the answers to a query that are consistent wrt a given set of ICs [1] (c.f. [3] for a survey). There, consistent data is characterized as invariant under all minimal restorations of consistency, i.e. as data that is present in all minimally repaired versions of the original instance (the *repairs*). In most of the research on CQA, a repair is a new instance that satisfies the given ICs, but differs from the original instance by a minimal set, under set inclusion, of (completely) deleted or inserted tuples. In that setting, changing the value of a particular attribute can be modelled as a deletion followed by an insertion, but this may not correspond to a minimal repair.

In certain applications, like those mentioned above, it may make more sense to consider correcting (updating) numerical values as a form of restoring consistency, which requires a new definition of repair that considers the quantitative nature of individual changes, the association of the numerical values to other key values and a quantitative distance between database instances.

Example 1. A company produces paper rolls and has a database with a relation *Roll* containing the *Id* of the rolls together with their grade and brightness. The rolls have grade 0 or 1 and the brightness takes integer val-

ues between 0 and 100. Rolls of grade 0 have brightness higher or equal to 88 and rolls of grade 1 have brightness higher or equal to 85 but smaller than 88. The following database D, with Id as the key, is inconsistent wrt this ICs. Under the tuple and set oriented semantics of repairs [1], the only repair corresponds to delete $Roll(2, 0, 86)$. However, we have two options that may make more sense than deleting the roll, namely changing the violating tuple to $Roll(2, 1, 86)$ or to $Roll(2, 0, 88)$. These options satisfy an implicit requirement that the numbers do not change too much. □

Roll	Id	Grade	Bright
	1	0	90
	2	0	86
	3	1	85

Update-based fixes for restoring consistency are studied in [13], however, the peculiarities of changing numerical attributes are not considered, and more importantly, the distance between databases instances used is, as in [1], set-theoretic, and not quantitative as in this paper. Those repaired versions in [13] are called *fixes*, and we keep this term (instead of *repairs*), because our basic repair actions are also changes of (numerical) attribute values.

In this paper we give a definition of a fix, we present some of complexity results related to fixing databases, some tractable cases, and an approximate solution for some hard cases. The presentation is rather informal, however rigorous definitions, theorems and proofs can be found in an extended version [4], where also many extensions can be found, in particular, aggregation constrains and aggregate queries are also considered, which is natural in this numerical context.

2. Preliminaries

The attributes of a database are called *flexible* if they take values in $\mathbb{Z}$ and are allowed to be fixed, and *hard* otherwise. $\mathcal{F}$ denoted the set of flexible attributes. Each relation in D is assumed to have a primary key that is satisfied by the initial instance D and by its fixes. In this sense, the primary key constraints are considered to be *hard* constraints. Attributes in a key are all hard. In addition, there may be a set of *flexible* ICs IC that may be violated, and it is the job of a fix to satisfy them again (while still satisfying the keys).

A *linear denial constraint* has the form $\forall \bar{x} \neg (A_1 \wedge \cdots \wedge A_m)$, with A_i a database atom, or a comparison $X \theta c$, where $\theta \in \{=, \neq, <, >, \leq, \geq\}$ (usually we replace $\wedge$ by commas).

Example 2. The constraints of example 1 can be expressed as linear denial constraints as follows: $\forall Id, G, B$ $\neg(Roll(Id, G, B), G > 1)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), G < 0)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), B < 0)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), B > 100)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), G = 0, B < 88)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), G = 1, B \geq 88)$, $\forall Id, G, B$ $\neg(Roll(Id, G, B), G = 1, B < 85)$. If the roll database also had a table Fr with the percentage of fiber in a roll, a constraint enforcing that rolls with grade 1 have a concentration of fiber ≥ 20 would be written: $\forall Id, G, B, F$ $\neg(Roll(Id, G, B), Fr(Id, F), G = 1, F < 20)$. □

3. Least Squares Fixes

When numerical values are updated to restore consistency, it is desirable to make the smallest overall variation wrt the original values while considering the relative relevance of the attributes. Since the original instance and fixes share key values, we can use them to compute numerical variations. For a tuple $\bar{t}_D$ in D, $\bar{t}_{D'}$ denotes the corresponding tuple in database D' that has the same key values as $\bar{t}_D$. To each attribute $A \in \mathcal{F}$ a numerical weight α_A is assigned.

The *square distance* between databases D and D' sharing the same schema and the same set of key values, is defined by $\Delta_{\bar{\alpha}}(D, D') = \sum_{\bar{t} \in D, A \in \mathcal{F}} \alpha_A (\pi_A(\bar{t}_D) - \pi_A(\bar{t}_{D'}))^2$, where π_A is the projection on A and $\bar{\alpha} = (\alpha_A)_{A \in \mathcal{F}}$. Accordingly, a *least squares fix* (LS-fix) for D wrt IC is defined as an instance D' such that (a) D' has the same schema and domain as D; (b) D' has the same values as D in the hard attributes (c) D' satisfies the key constraints and IC; and (d) minimizes the square distance $\Delta_{\bar{\alpha}}(D, D')$ over all the instances that satisfy (a) - (c).

Example 3. (example 1 continued) Given $\mathcal{F} = \{Grade, Bright\}$ and $\bar{\alpha} = (1, 1/8)$. Candidate fixes are $D_1 = \{(1, 0, 90), (2, 1, 86), (3, 1, 85)\}$ and $D_2 = \{(1, 0, 90), (2, 0, 88), (3, 1, 85)\}$, with $\Delta_{\bar{\alpha}}(D, D_1) = 1 \times 1^2$, and $\Delta_{\bar{\alpha}}(D, D_2) = 1/8 \times 2^2 = 0.5$. Then, D_2 is the only LS-fix. □

The coefficients α_A can be chosen in many ways, depending on their relevance, on the actual distribution of the data, or to compensate different scales of measurement. In this paper we will assume, for simplicity, that $\alpha_{A=1}$ for all $A \in \mathcal{F}$, and $\Delta_{\bar{\alpha}}(D, D')$ is simply denoted by $\Delta(D, D')$.

Example 4. A database D has tables $Client(Id, A, M)$, with key Id, attributes A for age and M for money; and $Buy(Id, I, P)$, with key $\{Id, I\}$, I for items, and P for prices. IC_1: $\forall Id, P, A, M \neg (Buy(Id, I, P), Client(Id, A, M), A < 18, P > 25)$ and IC_2: $\forall Id, A, M \neg (Client(Id, A, M), A < 18, M > 50)$, requiring that people younger than 18 cannot spend more

than 25 dollars on an item nor spend more than 50 dollars in the store. The following table is the inconsistent database D where the extra column is used to denote tuples.

Client			
Id	A	M	
1	15	52	t_1
2	16	51	t_2
3	60	90	t_3

Buy			
Id	I	P	
1	CD	27	t_4
1	TV	26	t_5
3	TV	40	t_6

IC_1 is violated by $\{t_1,t_4\}$ and $\{t_1,t_5\}$; and IC_2 by $\{t_1\}$ and $\{t_2\}$. There are two LS-fixes (the modified version of tuple t_1 is t_1', etc.), with $\Delta(D, D') = 2^2 + 1^2 + 2^2 + 1^2 = 10$, and $\Delta(D, D'') = 3^2 + 1^2 = 10$.

D':

Client'			
Id	A	M	
1	15	50	t_1'
2	16	50	t_2'
3	60	90	t_3

Buy'			
Id	I	P	
1	CD	25	t_4'
1	TV	25	t_5'
3	TV	40	t_6

D'':

Client"			
Id	A	M	
1	18	52	t_1''
2	16	50	t_2''
3	60	90	t_3

Buy"			
Id	I	P	
1	CD	27	t_4
1	TV	26	t_5
3	TV	40	t_6

Note that an LS-fix is not necessarily the result of applying "local" minimal fixes to tuples. □

The built-in atoms in linear denials determine an intersection of semi-spaces where the LS-fixes live. As shown in the previous example, they take values in the "borders" of the solution space [4]. It is not difficult to construct examples where an exponential number of fixes exists for a database. On the other side, for the kind of fixes and ICs we are considering, it is possible that no fix exists, in contrast to [1], where, if the set of ICs is logically consistent, a fix always exists.

Example 5. Relation $R(X, Y)$ has numerical attributes, X the key and Y flexible, and IC: $\forall X_1 X_2 Y \ \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 2)$, $\forall X_1 X_2 Y \ \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 3)$, $\forall X_1 X_2 Y \ \neg(R(X_1, Y), R(X_2, Y), X_1 = 2, X_2 = 3)$, $\forall XY \ \neg(R(X, Y), Y > 3), \forall XY \ \neg(R(X, Y), Y < 2)$. The first three ICs force Y to be different in every tuple. Last two ICs require $2 \leq Y \leq 3$. The database $R = \{(1, -1), (2, 1), (3, 5)\}$ has no fix. □

In applications where fixes are based on changes of numerical values, computing concrete fixes is a relevant problem. For example, correcting household forms in a census database before doing statistical processing is a common problem [8]. We can fix certain erroneous quantities as specified by the ICs. In these cases, the fixes are relevant objects to compute, which contrasts

with CQA [1], where the main motivation for introducing repairs is to characterize the notion of a consistent answer. In the following section we consider complexity and approximation for this problem.

4. Approximation for the Database Fix Problem

For a fixed set IC of linear denials, the problem $DFP(IC)$ of deciding whether there exists an LS-fix wrt IC at a distance $\leq k$, is NP-complete in data complexity [4]. The optimization problem $DFOP(IC)$ of finding the minimum distance from an LS-fix wrt IC to a given input instance, is $MAXSNP$-hard [4]; thus, unless $P = NP$, it has no *Polynomial Time Approximation Schema* [12].

Now, we restrict ourselves to a useful class of denial constraints, for which DFP is still NP-complete [4], but there is a good approximation algorithm. A set of linear denials IC is *local* if: (a) Attributes participating in equality atoms or joins are all hard; (b) There is a built-in with a flexible attribute in each IC; (c) No attribute A appears in IC both in comparisons of the form $A < c_1$ and $A > c_2$.[1] Local constraints have the property that there always exists an LS-fix, and it can be obtained by locally solving the initial inconsistencies, i.e. no new inconsistencies are be generated by those updates.

A *violation set* for $ic \in IC$ is a minimal set of tuples that simultaneously participate in the violation of ic. In example 4, where IC is local, they are $\{t_1,t_4\}$ and $\{t_1,t_5\}$ for IC_1 and $\{t_1\}$ and $\{t_2\}$ for IC_2. A *local fix* for $t \in D$ is a tuple t' that solves the inconsistency of at least one of the *violation sets* in which t is involved and minimizes the the quadratic distance $\Delta(\{t\}, \{t'\})$. Consistent tuples have no local fixes. The set $S(t, t')$ contains the violation sets that include t and are solved by changing t by t'. The $S(t, t')$ for example 4 are the columns in the table of example 6.

For a fixed set IC of local denials, we can solve an instance of $DFOP$ by transforming it into an instance of the *Minimum Weighted Set Cover Optimization Problem* ($MWSCP$) [10, Chapter 3] with elements in the underlying set U that have a bounded number of occurrences in the collection $\mathcal{S}$ of subsets of U. This will allow us to obtain a better approximation -within a constant factor- than for the general $MWSCP$, which is $MAXSNP$-hard [11, 12].

The underlying set U in the $MWSCP$ instance contains the violation sets for all $ic \in IC$. The set col-

---

1   To check the condition, replace $\leq c$ and $\geq c$ by $< c + 1$ and $> c - 1$, resp.

lection $\mathcal{S}$ for U contains the non-empty sets $S(t,t')$, where t' is a local fix for tuple $t \in D$, with weight $w(S(t,t')) = \Delta(\{t\}, \{t'\})$.

Example 6. (example 4 continued) We illustrate the reduction from *DFOP* to *MWSCP*. For the *MWSCP* instance, we need the local fixes. Tuple t_1 has two local fixes $t_1' = (1, 15, 50)$, that solves the violation set $\{t_1\}$ of IC_2, and $t_1'' = (1, 18, 52)$, that solves the violation sets $\{t_1, t_4\}$ and $\{t_1, t_5\}$ of IC_1, and $\{t_1\}$ of IC_2, with weights 4 and 9, resp. t_2, t_4 and t_5 have one local fix each corresponding to: $(2, 16, 50)$, $(1, CD, 25)$ and $(1, TV, 25)$ resp. The consistent t_3 has no local fix.

Set	S_1	S_2	S_3	S_4	S_5
Local Fix	t_1'	t_1''	t_2'	t_4'	t_5'
Weight	4	9	1	4	1
IC_1: $\{t_1, t_4\}$	0	1	0	1	0
IC_1: $\{t_1, t_5\}$	0	1	0	0	1
IC_2: $\{t_1\}$	1	1	0	0	0
IC_2: $\{t_2\}$	0	0	1	0	0

The *MWSCP* is shown in the table, where the elements (violation sets) are rows and the sets (e.g. $S_1 = S(t_1, t_1')$), columns. An entry 1 means that the set contains the corresponding element; and a 0, otherwise. □

If we solve instance $(U, \mathcal{S})$ for *MWSCP* by finding the minimum weight and a minimum weight cover $\mathcal{C}$, we could think of constructing a fix by replacing each inconsistent tuple $t \in D$ by a local fix t' with $S(t, t') \in \mathcal{C}$. The problem is that there might be more than one t' and the key dependencies would not be respected. For example, if we get $S(t, t_1)$ and $S(t, t_2)$ in $\mathcal{C}$, we would have two local fixes for t. It can be proved that there exists a set $S(t, t^\star) = S(t, t_1) \cup S(t, t_2)$ [4] and therefore we can obtain a fix in those cases by replacing t by $t^\star$. It can be proved that an instance $D(\mathcal{C})$ obtained from $\mathcal{C}$ in this way is an LS-fix that satisfies $w(\mathcal{C}) = \Delta(D, D(\mathcal{C}))$ [4]. It can also be proved that every LS-fix can be obtained in this way [4].

Example 7. (example 4 and 6 continued) We have two minimal covers $\mathcal{C}_1 = \{S_2, S_3\}$ and $\mathcal{C}_2 = \{S_1, S_3, S_4, S_5\}$ with weight 10. $D(\mathcal{C}_1)$ and $D(\mathcal{C}_2)$ correspond to the fixes of this problem. □

Using a greedy algorithm, *MWSCP* can be approximated within a factor $log(N)$, where N is the size of the underlying set U [7]. The approximation algorithm returns not only an approximation $\hat{w}$ to the optimal weight w^o, but also a cover $\hat{\mathcal{C}}$ (not necessarily optimal), which can be used to generate -as above- a database instance $D(\hat{\mathcal{C}})$ with the same key values as D that satisfies the constraints, but may not be LS-

minimal. It holds $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w} \leq log(N) \times w^o = log(N) \times \Delta(D, D')$, where D' is an LS-fix [4].

Example 8. (example 4 and 6 continued) We show how to use the approximation algorithm for *MWSCP* presented in [7] to compute an approximate solution to our *DFOP*. We start with $\hat{\mathcal{C}} := \emptyset$, $S_i^0 := S_i$; and we add to $\mathcal{C}$ the S_i such that S_i^0 has the maximum *contribution ratio* $|S_i^0|/w(S_i^0)$. Initially $|S_1|/w(S_1) = 1/4$, $|S_2|/w(S_2) = 3/9$, $|S_3|/w(S_3) = 1$, $|S_4|/w(S_4) = 1/4$ and $|S_5|/w(S_5) = 1$. The ratio is maximum for S_3 and S_5, so we can add any of them to $\hat{\mathcal{C}}$. If we choose the first, we get $\hat{\mathcal{C}} = \{S_3\}$. Now we compute the $S_i^1 := S_i^0 \smallsetminus S_3^0$, and choose again an S_i for $\hat{\mathcal{C}}$ such that S_i^1 maximizes the contribution ratio. Since S_5^1 gives the maximum it is added to $\hat{\mathcal{C}}$. By repeating this process until we get all the elements of U covered, i.e. all the S_i^k become empty at some iteration point, we obtain $\hat{\mathcal{C}} = \{S_3, S_5, S_1, S_4\}$. In this case $\hat{\mathcal{C}}$ is an optimal cover and $D(\hat{\mathcal{C}})$ is exactly the LS-fix D' in example 4. Since this is an approximation algorithm, in other examples the cover obtained might not be optimal. □

Thus, for local denials IC, we have a polynomial time approximation algorithm for $DFOP(IC)$ within a logarithmic factor, where N is the number of violation sets for D, which is polynomially bounded by the size of D [4]. However, in our instance of the cover problem the number of occurrences of elements of U in $\mathcal{S}$ is bounded by a constant that depends on the number of ICs, flexible attributes and atoms in the ICs [4]. If we apply the approximation algorithm in [10, Chapter 3] for this bounded set cover problem, we get an approximation within a constant factor [4] for any set of local constraints.

5. CQA Tractable Cases

The definition of a consistent answer to a query may depend upon different semantics, e.g. (a) *skeptical semantics*, where an answer is consistent if it is retrieved as a normal answer from *all* possible fixes; (b) *brave semantics*, where an answer is consistent if it is retrieved as a normal answer from *at least one* fix; (c) *majority semantics*, where an answer is consistent if it is obtained from most of the fixes; and (d) *range semantics*, where a consistent answer is the smallest range such that it contains the answers obtained from all the fixes [2]. The latter is a natural semantics for this kind of numerical problems. For all these semantics, consistent query answering is in general hard [4].

However, if we concentrate on the common class *1AD* of denials containing one database atom plus comparisons, e.g. $\forall Id, G, B \; \neg(Roll(Id, G, B), G = 0,$

$B < 88$), we can identify tractable cases for *CQA* under LS-fixes by reduction to *CQA* for (tuple and set-theoretic) repairs of the form introduced in [1] under key constraints. This is because each violation set contains one tuple, maybe with several local fixes, but all sharing the same key values; and then the problem consists in choosing one from different tuples with the same key values. The transformation preserves consistent answers to both ground and open queries. For tractability of *CQA* under LS-fixes, we can use results obtained in [9] for classical repairs.

The *join graph* $\mathcal{G}(Q)$ [9] of a conjunctive query Q is a directed graph, whose vertices are the literals in Q. There is an arc from L to L' if $L \neq L'$ and there is a variable w that occurs at the position of a non-key attribute in L and also occurs in L'. Furthermore, there is a self-loop at L if there is a variable that occurs at the position of a non-key attribute in L, and at least twice in L. If Q does not have repeated relations symbols, $Q \in \mathcal{C}_{Tree}$ if $\mathcal{G}(Q)$ is a forest and every non-key to key join of Q is full i.e. involves the whole key.

Using the reduction indicated above and results in [9], we get that for *1ADs* and queries in C_{Tree}, CQA under LS-fixes is in *PTIME* (in data complexity) under any of the semantics above [4]. However, for the same class of constraints and aggregate C_{Tree} queries, *CQA* is *NP*-hard under the brave semantics [4].

6. Conclusions

We have shown that fixing numerical values in databases that fail to satisfy some ICs, poses many new computational challenges not addressed before in the context of consistent query answering under classical repairs. In this paper we concentrated on integer values, which provide a natural and challenging domain. Extensions like moving to the real numbers would open completely different issues. Our framework could be applied to qualitative attributes with an implicit linear order given by the application.

Several extensions are reported in [4], and other decision problems are investigated. Complexity results, under different semantics, involve also aggregate conjunctive queries and aggregation constraints.

For related work, we refer to the literature on CQA (c.f. [3] for a survey and references). Papers [13] and [8] are the closest to our work, because changes in attribute values are basic repair actions, but the peculiarities of numerical values and quantitative distances between databases are not investigated. Under the set-theoretic, tuple-based semantics, [6, 5, 9] report on complexity issues for conjunctive queries, functional dependencies and foreign key constraints.

References

[1] Arenas, M., Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *Proc. ACM Symposium on Principles of Database Systems (PODS 99)*, 1999, pp. 68-79.

[2] Arenas, M., Bertossi, L. and Chomicki, J., He, X., Raghavan, V., and Spinrad, J. Scalar aggregation in inconsistent databases. *Theoretical Computer Science*, 2003, 296:405–434.

[3] Bertossi, L. and Chomicki, J. Query Answering in Inconsistent Databases. In 'Logics for Emerging Applications of Databases', J. Chomicki, G. Saake and R. van der Meyden (eds.), Springer, 2003.

[4] Bertossi, L., Bravo, L., Franconi, E. and Lopatenko, A. Fixing Numerical Attributes Under Integrity Constraints. CoRR paper cs.DB/0503032, arXiv.org e-Print archive.

[5] Cali, A., Lembo, D., Rosati, R. On the Decidability and Complexity of Query Answering over Inconsistent and Incomplete Databases. In *Proc. ACM Symposium on Principles of Database Systems (PODS 03)*, 2003, pp. 260-271.

[6] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, to appear.

[7] Chvatal, V. A Greedy Heuristic for the Set Covering Problem. *Mathematics of Operations Research*, 1979, 4:233-235.

[8] Franconi, E., Laureti Palma, A., Leone, N., Perri, S. and Scarcello, F. Census Data Repair: a Challenging Application of Disjunctive Logic Programming. In *Proc. Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 01)*. Springer LNCS 2250, 2001, pp. 561-578.

[9] Fuxman, A. and Miller, R. First-Order Query Rewriting for Inconsistent Databases. In *Proc. International Conference on Database Theory (ICDT 05)*, Springer LNCS 3363, 2004, pp. 337-354.

[10] Hochbaum, D. (ed.) *Approximation Algorithms for NP-Hard Problems*. PWS, 1997.

[11] Lund, C. and Yannakakis, M. On the Hardness of Approximating Minimization Problems. *Journal of the Association for Computing Machinery*, 1994, 45(5):960-981.

[12] Papadimitriou, Ch. *Computational Complexity*. Addison-Wesley, 1994.

[13] Wijsen, J. Condensed Representation of Database Repairs for Consistent Query Answering. In *Proc. International Conference on Database Theory (ICDT 03)*, Springer LNCS 2572, 2003, pp. 378-393.