# Fixing Numerical Attributes Under Integrity Constraints

**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 repairing databases by fixing numerical data at the attribute level. For their relevance in database applications with numerical domains, we consider denial and aggregate constraints. After providing a quantitative definition of database fix, we investigate the computational complexity of different problems, such as the existence and verification of fixes, and deciding consistency of answers to conjunctive aggregate queries. We show tractability for special cases; and provide approximation algorithms for some of the hard cases.

## 1 Introduction

Many database applications, like census, demographic, financial, and experimental data, contain quantitative data, usually associated to nominal or qualitative data, e.g. number of children associated to a household identification code (or address); or measurements associated to a sample identification code. It is common for this kind of 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 [9]. Other restrictions may be expressed by means of aggregation ICs, e.g. the maximum concentration of certain toxin in a sample may not exceed a certain specified amount; or the number of married men and married women must be the same. In this kind of applications, inconsistencies (or mistakes) 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) 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 and aggregation constraints; and conjunctive and conjunctive aggregate queries.

[*]Contact author. Phone: (1-613) 520-2600 x 1627. Fax: (1-613) 520-4334.
[†]Also: University of Manchester, Department of Computer Science, UK.

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 [2] (c.f. [4] 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 particular, an answer to a query is consistent when it can be obtained as a standard answer to the query from *every possible* repair. 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: (a) the quantitative nature of individual changes, (b) the association of the numerical values to other key values; and (c) a quantitative distance between database instances.

**Example 1.** A company pays at most $6,000$ to an employee with less than 5 years of experience. The following database $D$, with Name as the key, is inconsistent wrt this IC: Under the tuple and set oriented semantics of repairs [2], the only minimal repair corresponds to deleting the tuple $Employee(Robert, 4, 7000)$. However, we have two options that may make more sense that deleting the employee *Robert*, which is a value for the key, namely changing the violating tuple to $Employee(Robert, 5, 7000)$ or to $Employee(Robert,\ 4, 6000)$. These two options are satisfying an implicit requirement that the numbers do not change too much. □

| Employee | Name | Experience | Salary |
|---|---|---|---|
| | Sarah | 6 | 12,000 |
| | Robert | 4 | 7,000 |
| | Daniel | 5 | 8,000 |

Update based fixes for restoring the consistency of a database are studied in [20]; and changing values in attributes in a tuple is made a primitive repair action. In [20] semantic and computational problems around CQA are analyzed from the perspective of update based repairs. However, the peculiarities of changing numerical attributes are not considered, and more importantly, the distance between databases instances used in [20] is, as in [2], set-theoretic, and not quantitative, as we consider in this paper. Those repaired versions in [20] are called *fixes*, and we have decided to keep the same name (instead of *repairs*), because our basic repair actions are also changes of (numerical) attribute values.

The problem of correcting census data forms using disjunctive logic programs with stable model semantics is addressed in [9]. Several underlying assumptions that are necessary for the approach in [9] to work are made explicit and used in the current paper, extending the semantic framework introduced in [9].

In this paper we provide the semantic foundations for fixes that are based on changes on numerical attributes in the presence of key dependencies and wrt denial and aggregate integrity constraints. Alternative semantics are considered, and next the problem of CQA is reexamined accordingly. Fixing databases by changing numerical values while keeping the numerical distance to the original database to a minimum introduces interesting algorithmic and complexity theoretic issues. In consequence, we study decidability and complexity of different decision and optimization problems. We concentrate in particular on the "Database Fix Problem" (DFP), consisting in determining the existence of a fix at a distance not bigger than a given bound. We also consider the problems of construction and verification of such a fix. These problems are highly relevant for large inconsistent databases. For example, solving DFP can help us find the minimum distance from a fix to the original instance; information that can be used to prune impossible branches in the process of materialization of a fix.

We prove that DFP and CQA become undecidable in the presence of aggregation constraints. However, the DFP is NP-complete for linear denials, which are enough to capture census like applications. CQA belongs to $\Pi_2^P$ and becomes *coNP*-hard, but a relevant class of denials is identified for which CQA becomes tractable. Considering approximation algorithms, we prove that DFP is *MAXSNP*-hard [17], but can be approximated within a constant factor. All the algorithmic and complexity results, unless otherwise stated, refer to data complexity [1], i.e. to the size of the database that here includes a binary representation for numbers. All the proofs and some other technical results are in Appendix A.1.

## 2 Preliminaries

Consider a relational schema $\Sigma = (\mathcal{U}, \mathcal{R} \cup \mathcal{B}, \mathcal{A})$, where $\mathcal{U}$ is the database domain that includes $\mathbb{Z}$,[1] $\mathcal{R}$ is a set of database predicates, $\mathcal{B}$ a set of built-in predicates, and $\mathcal{A}$ is a set of attributes. A database instance is a finite collection $D$ of *database tuples*, i.e. of ground atoms of the form $P(\bar{c})$, where $P \in \mathcal{R}$ and $\bar{c}$ is a tuple of constants in $\mathcal{U}$. There is a set of attributes $\mathcal{F} \subseteq \mathcal{A}$ that contain all the *flexible* attributes, those that take values in $\mathbb{Z}$ and are allowed to be fixed. Attributes outside $\mathcal{F}$ are called *hard*. $\mathcal{F}$ need not contain all the numerical attributes.

We also have a set of key constraints $\mathcal{K}$, expressing that relations $R \in \mathcal{R}$ have a primary key $K_R$, $K_R \subseteq \mathcal{A}$. Later on (c.f. Definition 2), we will assume that $\mathcal{K}$ is satisfied both by an initial instance $D$, denoted $D \models \mathcal{K}$, and by its fixes. It also holds $\mathcal{F} \cap K_R = \emptyset$, i.e. values in key attributes cannot be changed in a fixing process; so the constraints in $\mathcal{K}$ are *hard*. In addition, there may be a separate set of *flexible* ICs *IC* that may be violated, and it is the job of a fix to satisfy them again (while still satisfying $\mathcal{K}$).

A *linear denial constraint* [14] has the form $\forall \bar{x} \neg(A_1, \ldots, A_m)$, where the $A_i$ are database atoms (i.e. with predicate in $\mathcal{R}$), or built-in atoms of the form $X\theta c$, where $\theta \in \{=, \neq, <, >, \leq, \geq\}$, and the commas represent conjunctions. We will also accept atoms of the form $X = Y$, that could be replaced by implicit equalities, i.e. by two different occurrences of the same variable in different database atoms. If we allow atoms of the form $X \neq Y$ in the denials, we will talk of *extended linear denials.*

**Example 2.** The following are linear denial constraints: (a) No customer is younger than 21: $\forall \mathit{Id}, \mathit{Age}, \mathit{Income}, \mathit{Status} \neg(\mathit{Customer}(\mathit{Id}, \mathit{Age}, \mathit{Income}, \mathit{Status}), \mathit{Age} < 21)$. (b) No customer with income less than 60000 has "silver" status: $\forall \mathit{Id}, \mathit{Age}, \mathit{Income}, \mathit{Status} \neg(\mathit{Customer}(\mathit{Id}, \mathit{Age}, \mathit{Income}, \mathit{Status}), \mathit{Income} < 60000, \mathit{Status} = \mathit{silver})$. (c) The reference person in a household and his/her spouse must have different sex: $\forall \mathit{Address}, \mathit{Status1}, \mathit{Status2}, \mathit{Sex1}, \mathit{Sex2} \neg(\mathit{Household}(\mathit{Address}, \mathit{Status1}, \mathit{Sex1}), \mathit{Household}(\mathit{Address}, \mathit{Status2}, \mathit{Sex2}), \mathit{Status1} = \mathit{reference}, \mathit{Status2} = \mathit{spouse}, \mathit{Sex1} = \mathit{Sex2})$. □

In this paper, in order to present some decidability and complexity-theoretic issues, we will only consider the aggregation function *sum*, which can be used in some classes of aggregation constraints (ACs) and aggregation queries. For a detailed account of ACs we refer to [19]. *Filtering* ACs are obtained by imposing conditions on the set of tuples over *sum* is applied, e.g. $\mathit{sum}(A_1\colon A_2 = 3) > 5$ refers the sum over $A_1$ of tuples with $A_2 = 3$. *Multi-attribute* ACs allow arithmetical combinations of attributes as arguments for *sum*, e.g. $\mathit{sum}(A_1 + A_2) > 5$ and $\mathit{sum}(A_1 \times A_2) > 100$. If an AC involves attributes from more than one relation, we call it *multi-relation*, e.g. $\mathit{sum}_{R_1}(A_1) = \mathit{sum}_{R_2}(A_1)$, otherwise it is *single-relation.*

An *aggregate conjunctive query* has the form $q(x_1, \ldots x_m;\ \mathit{sum}(z)) \leftarrow B(x_1, \ldots, x_m, z, y_1, \ldots, y_n)$, where its *non-aggregate matrix* (NAM) given by $q'(x_1, \ldots x_m) \leftarrow B(x_1, \ldots, x_m,$

[1]If necessary, with simple denial constraints, numbers can be restricted to $\mathbb{N}$, or any interval, e.g. $\{0, 1\}$.

$z, y_1, \ldots, y_n)$ is a usual first-order (FO) conjunctive query with built-in atoms, and the $z$ does not appear among the $x_i$. Here the *aggregation attribute* is $z$. An aggregate conjunctive query is *cyclic* (*acyclic*) if its NAM is a cyclic (acyclic) query [1]. An *aggregate comparison query* is a sentence of the form $q(sum(z)), sum(z)\theta k$, where $q(sum(z))$ is the head of an aggregate conjunctive query (with no free variables), $\theta$ is a comparison operator, and $k$ is an integer number. For example, $Q\colon q(sum(z)), sum(z) > 5$, with $q(sum(z)) \leftarrow R(x,y), Q(y,z,w), w \neq 3$ is an aggregate comparison query.

## 3 Least Squares Fixes

When we update numerical values to restore consistency, it is desirable to make the smallest overall variation of the original values. Since the original instance and a fix share the same key values, we can use them to compute variations in the numerical values. For a tuple $\bar{k}$ of values for the key $K_R$ of relation $R$ in an instance $D$, $\bar{t}(\bar{k}, R, D)$ denotes the unique tuple $\bar{t}$ in relation $R$ in instance $D$ whose key value is $\bar{k}$.

**Definition 1.** For instances $D$ and $D'$ over schema $\Sigma$ with the same set $val(K_R)$ of tuples of key values for each relation $R \in \mathcal{R}$, their *square distance* is $\Delta(D, D') = \sum_{\substack{R\in\mathcal{R}, A\in\mathcal{F}\\ \bar{k}\in val(K_R)}} (\pi_A(\bar{t}(\bar{k}, R, D)) - \pi_A(\bar{t}(\bar{k}, R, D')))^2$, where $\pi_A$ is the projection on attribute $A$. □

**Definition 2.** For an instance $D$, a set of flexible attributes $\mathcal{F}$, a set of key dependencies $\mathcal{K}$, such that $D \models \mathcal{K}$, and a set of flexible ICs $IC$, a *least squares fix* (LS-fix) for $D$ wrt $IC$ is 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 attributes in $\mathcal{A} \smallsetminus \mathcal{F}$; (c) $D' \models \mathcal{K}$; (d) $D' \models IC$; and (e) minimizes the square distance $\Delta(D, D')$ over all the instances that satisfy (a) - (d). □

**Example 3.** (example 1 cont.) Here, $\mathcal{R} = \{Employee\}$, $\mathcal{A} = \{Name, Experience, Salary\}$, $\mathcal{F} = \{Experience, Salary\}, K_{Employee} = \{Name\}$. For $D$, the original instance, $val(K_{Employee})$ $= \{Sarah, Robert, Daniel\}, \bar{t}(Sarah, Employee, D) = (Sarah, 6, 12000)$, etc. Candidate for fixes were: $D_1 = \{ (Sarah, 6, 12{,}000), (Robert, 5, 7{,}000), (Daniel, 5, 8{,}000)\}$, $D_2 = \{ (Sarah, 6, 12{,}000), (Robert, 4, 6{,}000), (Daniel, 5, 8{,}000)\}$, with square distances to $D$: $\Delta(D, D_1) = 1$, and $\Delta(D, D_2) = 10^6$, resp. $D_1$ is the only LS-fix. □

**Example 4.** Database $D$ has tables $Client(ID, A, M)$, with key $Id$, attributes $A$ for age and $M$ for amount of money; and $Buy(ID, I, P)$, with key $\{ID, I\}$, $I$ for items, and $P$ for prices. We have denials $IC_1\colon \forall ID, P, A, M \neg( Buy(ID, I, P), Client(ID, A, M), A < 18, P > 25)$ and $IC_2\colon \forall ID, A, M \neg( Client( ID, A, M), A < 18, M > 50)$, requiring that people younger than 18 cannot spend more than 25 on one item nor spend more than 50 in the store. We added an extra column in the tables with a notation for each tuple. $IC_1$ is violated by $\{t_1, t_4\}$ and $\{t_1, t_5\}$; and $IC_2$ by $\{t_1\}$ and $\{t_2\}$.

$D$:

| **Client** | **ID** | **A** | **M** | |
|---|---|---|---|---|
| | 1 | 15 | 52 | $t_1$ |
| | 2 | 16 | 51 | $t_2$ |
| | 3 | 60 | 900 | $t_3$ |
| **Buy** | **ID** | **I** | **P** | |
| | 1 | CD | 27 | $t_4$ |
| | 1 | DVD | 26 | $t_5$ |
| | 3 | DVD | 40 | $t_6$ |

$D'$:

| **Client'** | **ID** | **A** | **M** | |
|---|---|---|---|---|
| | 1 | 15 | 50 | $t_1'$ |
| | 2 | 16 | 50 | $t_2{}'$ |
| | 3 | 60 | 900 | $t_3$ |
| **Buy'** | **ID** | **I** | **P** | |
| | 1 | CD | 25 | $t_4{}'$ |
| | 1 | DVD | 25 | $t_5{}'$ |
| | 3 | DVD | 40 | $t_6$ |

$D''$:

| **Client"** | **ID** | **A** | **M** | |
|---|---|---|---|---|
| | 1 | 18 | 52 | $t_1{}''$ |
| | 2 | 16 | 50 | $t_2{}''$ |
| | 3 | 60 | 900 | $t_3$ |
| **Buy"** | **ID** | **I** | **P** | |
| | 1 | CD | 27 | $t_4$ |
| | 1 | DVD | 26 | $t_5$ |
| | 3 | DVD | 40 | $t_6$ |

We have 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 $\Delta(D, D'') = 3^2 + 1^2 = 10$, resp. We can see that a global 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 (c.f. Proposition A.1, Appendix A.1). 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 [2, 3], where, if the set of ICs is consistent as a set of logical sentences, it always exists a fix for a database.

**Example 5.** Relation $R(X, Y)$ has numerical attributes, $X$ the key and $Y$ flexible, and $IC_1$: $\forall X_1X_2Y\neg(R(X_1, Y), R(X_2, Y), X_1{=}1, X_2{=}2), \forall X_1X_2Y\neg(R(X_1, Y), R(X_2, Y), X_1{=}1, X_2{=}3),$ $\forall X_1X_2Y\neg(R(X_1, Y), R(X_2, Y), X_1{=}2, X_2{=}3), \forall XY\neg(R(X, Y), Y{>}3), \forall XY\neg(R(X, Y), Y{<}2)$. The first three ICs force $Y$ to be different in every tuple. Last two ICs require $2 \leq Y \leq 3$. The inconsistent database $R = \{(1, -1), (2, 1), (3, 5)\}$ has no fix. Now, with $IC_2$ containing $\forall X, Y\neg($ $R(X, Y)$, $Y > 1)$ and $\mathit{sum}(Y) = 10$, any database with less than 10 tuples has no fixes. □

In applications where fixes are based on changes of numerical values, computing concrete fixes is a relevant problem. For example, in databases containing census forms, correcting the latter before doing statistical processing is a common problem [9]. In databases with experimental samples, we can fix certain erroneous quantities as specified by linear ICs. In these cases, the fixes are relevant objects to compute explicitly, which contrasts with CQA [2], where the main motivation for introducing repairs is to formally characterize the notion of a consistent answer to a query as an answer that remains under all possible fixes. In consequence, we now consider some decision problems related to existence and verification of fixes, and to CQA under different semantics, whose suitability may be depend on the application.

**Definition 3.** For an instance $D$ and set of ICs $IC$, we denote (a) $\mathit{Fix}(D, IC) := \{D' \mid D'$ is an LS-fix of $D$ wrt $IC\}$. (b) $\mathit{Fix}(IC) := \{(D, D') \mid D' \in \mathit{Fix}(D, IC)\}$. (c) $\mathit{NE}(IC) := \{D \mid \mathit{Fix}(D, IC) \neq \emptyset\}$, for *non-empty* set of fixes. (d) $\mathit{NE} := \{(D, IC) \mid \mathit{Fix}(D, IC) \neq \emptyset\}$. (e) $\mathit{DFP}(IC) {:=} \{(D, k) |$ there is $D' \in \mathit{Fix}(D, IC)$ with $\Delta(D, D') \leq k\}$, the *database fix problem.* □

**Definition 4.** Let $D$ be a database, $IC$ a set ICs, and $Q$ a conjunctive query. (a) A ground tuple $\bar{t}$ is a *consistent answer* to $Q$ under the: (a1) *skeptical semantics* if for every $D' \in \mathit{Fix}(D, IC)$, $D' \models Q(\bar{t})$. (a2) *brave semantics* if there exists $D' \in \mathit{Fix}(D, IC)$ with $D' \models Q(\bar{t})$. (a3) *majority semantics* if $|\{D' \mid D' \in \mathit{Fix}(D, IC)$ and $D' \models Q(\bar{t})\}| > |\{D' \mid D' \in \mathit{Fix}(D, IC)$ and $D' \not\models Q(\bar{t})\}|$. (b) $\mathit{Cqa}(Q, D, IC, \mathcal{S})$ is the set of consistent answers to $Q$ in $D$ wrt $IC$ under semantics $\mathcal{S}$. If $Q$ is ground, $\mathit{Cqa}(Q, D, IC, \mathcal{S}) := \{\mathit{yes}\}$ when $D \models_{\mathcal{S}} Q$, i.e. is true in the fixes of $D$ according to semantics $S$; otherwise $\mathit{Cqa}(Q, D, IC, \mathcal{S}) := \{\mathit{no}\}$. (c) $\mathit{Cqa}(Q, IC, \mathcal{S}) := \{(D, \bar{t}) \mid \bar{t} \in \mathit{Cqa}(Q, D, IC, \mathcal{S})\}$, the *problem of consistent query answering.* □

## 4 Decidability and Complexity

**Theorem 1.** The problem $\mathit{NE}$, of existence of LS-fixes, under extended linear denials and complex, filtering, multi-attribute, single-relation, aggregation constraints is undecidable. □

This result can be proved by reduction from de undecidable Hilbert's problem on solvability of diophantine equations (c.f. Appendix A.2 for an example). Here we have the original database and the set of ICs as input parameters.

**Lemma 1.** $\mathit{NE}(IC)$ can be reduced in polynomial time to the complements of $\mathit{Cqa}(\mathit{False}, IC,$ $\mathit{Skeptical})$, $\mathit{Cqa}(\mathit{True}, IC, \mathit{Majority})$, where $\mathit{False}$, $\mathit{True}$ are always false, resp. true. □

In Lemma 1, it suffices for queries *False*, *True* to be false, resp. true, in all instances that share the key values with the input database. Then, they can be represented by $\exists Y R(\bar{c}, Y)$, where $\bar{c}$ are not (for *False*), or are (for *True*) key values in the original instance.

**Corollary 1.** Under the hypothesis of Theorem 1, CQA under the skeptical or majority semantics is undecidable. □

In the following we will be interested in data complexity, when only the input database varies and the set of ICs is fixed [1].

**Lemma 2.** For an instance $D$ if there is a database instance $D'$ with the same schema and key values as $D$ that satisfies *IC*, then there is an LS-fix of $D$ wrt *IC*. □

Example 5, whose ICs ar logically consistent, does not satisfy the hypothesis of this lemma.

**Proposition 1.** For a fixed set *IC* of linear denials, deciding if an instance $D$ has an instance $D'$ (with the same key values as $D$) that satisfies *IC* with $\Delta(D, D') \leq k$, $k$ an integer number, is in *NP*. □

By Lemma 2, there is $D'$ with the same key values as $D$ that satisfies the ICs at a distance $\leq k$ iff there is an LS-fix at a distance $\leq k$. Thus, with Proposition 1 and a reduction of the *Vertex Cover Problem* to $DFP(IC_0)$ for a fixed set of denials $IC_0$, we obtain:

**Theorem 2.** Under the hypothesis of Proposition 1, *DFP*(*IC*), the problem of deciding whether there exists an LS-fix wrt *IC* at a distance $\leq k$, is *NP*-complete. □

By Proposition 1, if there is a fix at a distance $\leq k$, the minimum distance to $D$ for a fix can be found by binary search in $log(k)$ steps. Actually, if an LS-fix exists, its square distance to $D$ is polynomially bounded by the size of $D$ (c.f. proof of Theorem 3). Since $D$ and a fix have the same number of tuples, what matters is the size of the values taken in a fix, which are constrained by a fixed set of linear denials and the condition of minimality.

**Theorem 3.** For a fixed set *IC* of extended linear denials, the problem *NE*(*IC*), of deciding if an instance has an LS-fix wrt *IC*, is *NP*-complete. □

**Corollary 2.** Under the hypothesis of Theorem 3, CQA under the skeptical and the majority semantics is *coNP*-hard. □

**Theorem 4.** For extended linear denials, the problem *Fix*(*IC*) of checking if an instance is an LS-fix is *coNP*-complete. □

**Corollary 3.** For extended linear denials, CQA under skeptical semantics is in $\Pi^P_2$. [17] □

The *Vertex Cover Problem* can be reduced in polynomial time to CQA for aggregate comparison queries under the brave semantics:

**Proposition 2.** Consistent query answering for aggregate comparison queries under linear denials and brave semantics is *coNP*-hard. □

## 5 Approximation

*DFOP*(*IC*) is the problem of finding the minimum distance from an LS-fix to the given instance. There is an $L$-reduction to *DFOP*(*IC*) from the *MAXSNP*-complete [18, 17] *B-Minimum Vertex Cover Problem*, the minimization vertex cover problem for graphs of bounded degree [13, Chapter 10]. Thus, unless $P = NP$, there is no *Polynomial Time Approximation Schema* for *DFOP*, in particular, it cannot be uniformly approximated within arbitrarily small constant factors [17].

**Proposition 3.** For a fixed set of linear denials *DFOP* is *MAXSNP*-hard. $\square$

Now we present an approximation algorithm for a restricted class of useful denial constraints, that have the property that by doing local fixes, no new inconsistencies will be generated and there will always be an LS-fix wrt to them (c.f. Proposition A.3, Appendix A.1).

**Definition 5.** A set of linear denials *IC* is *local* if: (a) Equalities between attributes and joins involve only hard attributes; (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 $A > c_2$. [2] $\square$

In Example 5, *IC* is not local. Locality is a sufficient (c.f. Proposition A.3, Appendix A.1), but not necessary condition for existence of LS-fixes. In Example 4, *IC* is local. For local denials *DFP* is still *NP*-complete (cf. Proposition A.2, Appendix A.1).

**Definition 6.** A set $I$ of database tuples from $D$ is a *violation set* for an IC $ic$ if $I \not\models ic$, and for every $I' \subsetneqq I$, $I' \models ic$. $\mathcal{I}(D, ic, t)$ is the set of violation sets for $ic$ that contain $t$. A tuple $t$ is *consistent* if $\mathcal{I}(D, ic, t) = \emptyset$ for every $ic \in IC$. $\square$

A violation set $I$ for $ic$ is a minimal set of tuples that simultaneously participate in the violation $ic$. We label $I$ with the corresponding $ic$ using the pair $(I, ic)$.

**Definition 7.** Given an instance $D$ and ICs *IC*, a *local fix* for $t \in D$, is a tuple $t'$ with: (a) the same values for the hard attributes as $t$; (b) $S(t,t') := \{(I, ic) \mid ic \in IC,\ I \in \mathcal{I}(D, ic, t)$ and $((I \smallsetminus \{t\}) \cup \{t'\}) \models ic\} \neq \emptyset$; and (c) there is no tuple $t''$ that simultaneously satisfies (a), $S(t, t'') = S(t, t')$, and $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$. $\square$

$S(t, t')$ contains the violation sets that include $t$ and are solved changing $t$ by $t'$. A local fix $t'$ solves some of them and minimizes the distance to $t$. Consistent tuples have no local fixes.

For a fixed set *IC* of local denials, we can solve an instance of *DFOP* by transforming it into an instance of the *MAXSNP*-hard *Minimum Weighted Set Cover Optimization Problem* (*MWSCP*) [16, 17]. By concentrating on local denials, we will obtain an instance of *MWSCP* 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 better approximation results than for the general case of *MWSCP* [16], namely a constant approximation factor.

Given $D$, consider the *conflict hyper-graph* $\mathcal{G}$ [7], whose vertices are the database tuples, and hyper-edges are the violation sets for elements $ic \in IC$ (labelled with the corresponding $ic$, so that we can have different hyper-edges with the same tuples in them). The underlying set $U$ in the instance for *MWSCP* contains the hyper-edges of $\mathcal{G}$. The set collection $\mathcal{S}$ for $U$ contains the non-empty sets $S(t, t')$, where $t'$ is a local fix for tuple $t \in D$, with weight $w(S(t, t')) = \Delta(\{t\}, \{t'\})$ (by Proposition A.3, Appendix A.1, $\mathcal{S}$ covers $U$).

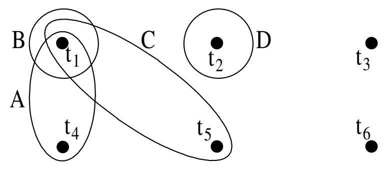
If we solve instance $(U, \mathcal{S})$ for *MWSCP* by finding the minimum weight and a minimum weight cover $\mathcal{C}$, we could think of constructing a fix by replacing each inconsistent tuple $t \in D$ by a local fix $t'$ with $S(t, t') \in \mathcal{C}$. The problem is that there might be more than one $t'$ and the key dependencies are not respected. However, we obtain an LS-fix $D(\mathcal{C})$ as follows: (1) For each tuple $t$ with local fixes $t_1, \ldots, t_n$, $n > 1$, and $S(t, t_i) \in \mathcal{C}$, obtain $\mathcal{C}'$ replacing in $\mathcal{C}$ all the $S(t, t_i)$ by $S(t, t^\star)$, where $t^\star$ is such that $S(t, t^\star) = \bigcup_{i=1}^{n} S(t, t_i)$; (2) Let $D(\mathcal{C})$ be the instance obtained from $D$ replacing $t$ by $t'$ if $S(t, t') \in \mathcal{C}'$. It holds that $D(\mathcal{C})$ is an LS-fix, and $w(\mathcal{C}) = w(\mathcal{C}') = \Delta(D, D(\mathcal{C}))$, so that the value for the objective function is kept. (Propositions A.4 and A.5, Appendix A.1 prove all the last claims and the fact that we find $S(t, t^\star) \in \mathcal{S}$.)

It can also be proved that every LS-fix can be obtained in this way (c.f. Proposition A.6, Appendix A.1). We notice that the transformation from DFOP to MWSCP, and the construc-

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

tion of $D(\mathcal{C})$ from $\mathcal{C}$ can both be done in polynomial time in the size of $D$ (c.f. Proposition A.7, Appendix A.1).

**Example 6.** (example 4 continued) We illustrate the reduction from *DFOP* to *MWSCP*. The violation sets are $\{t_1,t_4\}$ and $\{t_1,t_5\}$ for $IC_1$ and $\{t_1\}$ and $\{t_2\}$ for $IC_2$. The figure shows the hyper-graph. For the *MWSCP* instance, we need the local fixes. Tuple $t_1$ has one local fix wrt $IC_1$ and one wrt $IC_2$, with weights 4 and 9, resp. Tuple $t_2$ has one local fix with weight 1: changing $M$ from 51 to 50. Tuples $t_4$ and $t_5$ have one local fix each, with weights 4 and 1, resp.



| Set | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| Local Fix<br>Weight | $t_1$'<br>4 | $t_1$"<br>9 | $t_2$'<br>1 | $t_4$'<br>4 | $t_5$'<br>1 |
| Hyperedge A | 0 | 1 | 0 | 1 | 0 |
| Hyperedge B | 1 | 1 | 0 | 0 | 0 |
| Hyperedge C | 0 | 1 | 0 | 0 | 1 |
| Hyperedge D | 0 | 0 | 1 | 0 | 0 |

The *MWSCP* instance is shown in the table, where the elements are rows and the sets, columns. An entry 1 means that the set contains the corresponding element; and a 0, otherwise. There are two minimal covers, both with weight 10: $\mathcal{C}_1 = \{S_2, S_3\}$ and $\mathcal{C}_2 = \{S_1, S_3, S_4, S_5\}$. $D(\mathcal{C}_1)$ and $D(\mathcal{C}_2)$ are the two fixes for this problem. □

If we apply the transformation to Example 5, which has a non-local set of ICs, we will find that instance $D(\mathcal{C})$, for $\mathcal{C}$ a set cover, can be constructed as above, but it does not satisfy the flexible ICs, because changing inconsistent tuples by their local fixes solves only the initial inconsistencies, but new inconsistencies are introduced.

Using a greedy algorithm, *MWSCP* can be approximated within a factor $log(N)$, where $N$ is the size of the underlying set $U$ [8]. The approximation algorithm returns not only an approximation $\hat{w}$ to the optimal weight $w^\star$, but also a cover $\hat{\mathcal{C}}$ (not necessarily optimal). This cover can be used to generate, via $(\hat{\mathcal{C}})'$ as before, a database instance $D(\hat{\mathcal{C}})$ with the same key values as $D$ that satisfies $\mathcal{K} \cup IC$, but may not be LS-minimal. It holds $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w}$ (c.f. Proposition A.8, Appendix A.1). In consequence, $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w} \leq log(N) \times w^\star = log(N) \times \Delta(D, D')$, where $D'$ is an LS-fix.

In consequence, for any set $IC$ of local denials, we have a polynomial time approximation algorithm that solves $DFOP(IC)$ within an $O(log(N))$ factor, where $N$ is the number of violation sets for $D$ wrt $IC$. This number $N$, the number of hyper-edges in $\mathcal{G}$, is polynomially bounded by the $|D|$ (c.f. Proposition A.7, Appendix A.1), and may be relatively small if the number of inconsistencies is small or the number of database atoms in the ICs is small, which is likely the case in real applications. However, if we apply approximation algorithms for the special case of *MWSCP* where the number of occurrences of an element of $U$ in elements of $\mathcal{S}$ is bounded by a constant, we can get an approximation within a constant factor [13, Chapter 3]. This is clearly the case in our application, being $m \times |\mathcal{F}| \times |IC|$ a constant -and $|D|$ independent- bound on the frequency of the elements, where $m$ is the maximum number of database atoms in an IC.

## 6 One Database Atoms Denials

We concentrate on *one database atom denials* (1DAD), i.e. of the form $\forall \neg(A, B)$, where atom $A$ has a predicate in $\mathcal{R}$, and $B$ is a conjunction of built-in atoms. They capture range constraints; and census data is usually stored in single relation schemas [9]. For 1DADs, we can identify tractable cases for CQA by reducing CQA under LS-fixes to CQA for (tuple and set-theoretic) repairs of the form introduced in [2] for 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 (c.f. proof of Theorem 5). The transformation preserves answers to ground and open queries. The "classical" repair problem has been studied in detail in [7, 6, 10]. For tractability of CQA, we can use results about and algorithms for CQA obtained in [10] for the classical framework.

The *join graph* $\mathcal{G}(Q)$ [10] of a conjunctive query $Q$ is a directed graph, whose vertices are the database atoms 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 $L$. When $Q$ does not have repeated relations symbols, we write $Q \in \mathcal{C}_{\mathit{Tree}}$ if $\mathcal{G}(Q)$ is a forest and every non-key to key join of $Q$ is full i.e. involves the whole key. $\mathcal{C}_{\mathit{Tree}}$ provides an extreme case of tractability of classical CQA, i.e. relaxing the conditions on the queries we get intractability [10].

**Theorem 5.** *For a fixed set of 1DADs and queries in $C_{\mathit{Tree}}$, consistent query answering is in PTIME.* $\square$

A conjunctive aggregate query belongs to $C_{\mathit{Tree}}$ if its underlying non-aggregate conjunctive query (its NAM) belongs to $C_{\mathit{Tree}}$. Even for 1DADs, with simple comparison aggregate queries with *sum*, tractability is lost under the brave semantics.

**Proposition 4.** For a fixed set of 1DADs, CQA for aggregate queries that are in $C_{\mathit{Tree}}$ or acyclic is *NP*-hard under the brave semantics. $\square$

For queries $Q$ that return a numerical value, the *range semantics* for CQA provides the *min-max* and *max-min* answers, i.e. the supremum and the infimum, resp., of the set of answers to $Q$ from the fixes [3]. Using the *Bounded Degree Independent Set Problem* [12] we obtain:

**Proposition 5.** There is a fixed set of 1DADs and a fixed aggregate conjunctive query, such that CQA under range semantics is *NP*-hard. $\square$

## 7 Conclusions

We have shown that fixing numerical values in databases that fail to satisfy some integrity constraints poses many new computational challenges that had not been addressed before in the context of consistent query answering. In this paper we have just started to investigate some of the many problems that appear in this context. We concentrated on integer values, which provide a useful and challenging domain. Moving to the real numbers would open many new issues. Our framework could be applied to qualitative attributes with an implicit linear order given by the application. We have developed (but not reported here) extensions to our approach that consider *minimum distribution variation* LS-fixes that keep the overall statistical properties of the database, and *weighted LS-fixes* that capture the relative relevance of numerical attributes and scales of measurement issues.

The *range semantics* for CQA for queries that return numbers should be further investigated. It was introduced in [3] for aggregate queries under functional dependencies and set-theoretic, tuple-based repairs. There, a minimal interval is returned that contains the individual numerical answers from all the possible fixes. Also more research on the role of aggregation constraints is needed.

For related work, we refer to the literature on CQA (c.f. [4] for a survey and references). Papers [20] and [9] 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, [7, 5, 10] report on complexity issues for conjunctive queries, functional dependencies and foreign key constraints. A majority semantics was studied in [15] for database merging.

**Acknowledgments:** Research partially supported by NSERC (Grant 250279-02), CITO/IBM-CAS Student Internship Program, and EU projects: Sewasie, Knowledge Web, and Interop. L. Bertossi is Faculty Fellow of IBM Center for Advanced Studies (Toronto). The first two authors are grateful to Alberto Mendelzon for stimulating conversations at a very early stage of this research.

## References

[1] Abiteboul, S., Hull, R. and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.

[2] 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.

[3] 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.

[4] 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.

[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., Marcinkowski, J. and Staworko, S. Computing Consistent Query Answers Using Conflict Hypergraphs. In *Proc. 13th ACM Conference on Information and Knowledge Management (CIKM 04)*, ACM Press, 2004, pp. 417-426.

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

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

[9] 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.

[10] Fuxman, A. and Miller, R. First-Order Query Rewriting for Inconsistent Databases. In *Proc. International Conference on Database Theory (ICDT 05)*, to appear.

[11] Garey, M.R. and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., 1979.

[12] Garey, M., Johnson, D. and Stockmeyer, L. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1976, 1(3):237-267.

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

[14] Kuper, G., Libkin, L. and Paredaens, J. (eds.) *Constraint Databases.* Springer, 2000.

[15] Lin, J. and Mendelzon, A.O. Merging Databases under Constraints. *International Journal of Cooperative Information Systems*, 1996, 7(1):55-76.

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

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

[18] Papadimitriou, Ch. and Yannakakis, M. Optimization, Approximation and Complexity Classes. *J. Computer and Systems Sciences*, 1991, 43:425-440.

[19] Ross, K., Srivastava, D., Stuckey, P., and Sudarshan, S.. Foundations of Aggregation Constraints. *Theoretical Computer Science*, 1998, 193(1-2):149–179.

[20] 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.

# A Appendix

## A.1 Proofs

Those auxiliary technical results that are stated in this appendix, but not in the main body of the paper, are numbered in the form **A.n**, e.g. Lemma A.1.

The following lemma proves that if a tuple is involved in an inconsistency, the set of constraints is consistent and there is at least one flexible attribute in each integrity constraint, then there always exists a local fix for it.

**Lemma A.1.** For a database $D$ and a consistent set of linear denial constraints $IC$, where each constraint contains at least one built-in involving a flexible constraint and there are equalities or joins only between hard attributes. Then, for every tuple $t$ with at least one flexible attribute and at least one $ic$ in $IC$, $\mathcal{I}(D, ic, t) \neq \emptyset$, there exists at least one local fix $t'$ $\square$

**Proof:** Each constraint $ic \in IC$ has the form $\forall \bar{x} \neg(P_1(\bar{x}), \ldots, P_n(\bar{x}), A_i < c_i, A_j \geq c_j, A_k = c_k, A_l \neq c_l, \ldots)$ and can be rewritten as a clause only with $<$, $>$ and $=$:

$$\forall \bar{x}(\neg P_1(\bar{x}) \vee \ldots \vee \neg P_n(\bar{x}) \vee A_i \geq c_i \vee A_j < c_j \vee A_k < c_k \vee A_k > c_k \vee A_l = c_l \vee \ldots) \quad (1)$$

This formula shows that since the repairs are done by attributes updates, the only way we have of solving an inconsistency is by fixing at least one of the values of a flexible attribute. Let $ic$ be a constraint in $IC$ such that $\mathcal{I}(D, ic, t) \neq \emptyset$ and $I$ be a violation set $I \in \mathcal{I}(D, ic, t)$. Now, since $ic \in IC$, $ic$ is a consistent constraints. Then for each flexible attribute $A$ in $ic$ we are able to derive an interval $[c_l, c_u]$ such that if the value of $A$ is in it, we would restore the consistency of $I$. For example if we have a constraint in form of equation (1) with $A \leq 5$, then, if we want to restore consistency by modifying $A$ we would need to have $A \in (-\infty, 5]$. If the constraint had also $A \geq 1$ the interval would be $[1, 5]$. Since $t$ has at least one flexible attribute and each flexible attribute has an interval, it is always possible to adjust the value of that flexible attribute to a value in the interval $[c_l, c_u]$ and restore consistency. By finding the adjustment that minimizes the distance from the original tuple we have find a local fix for the tuple $t$. $\square$

The *borders* of an attribute in an extended denial correspond to the surfaces of the semi-spaces determined by the buil-in atoms in it.

**Proposition A.1.** Given a database $D$ and a set of linear denials $IC$, where equalities and joins can only exist between hard attributes, the values in every flexible attributes in a local fix $t'$ (c.f. Definition 7) of a tuple $t \in D$ will correspond to the original value in $t$ or to a border of a constraint in $IC$. Furthermore, the values in every attributes of a tuple $t' \in D'$ will correspond to the original value of the attribute in the tuple in $D$ or to a border of a constraint in $IC$. $\square$

**Proof:** First we will replace in all the constraints $X \leq c$ by $X < (c+1)$, $X \geq c$ by $X > (c-1)$ and $X = c$ by $(X > (c-1) \wedge X < (c+1))$. We can do this because we are dealing with integer values. Then, a constraint $ic$ would have the form $\forall \bar{x} \neg(P_1(\bar{x}), \ldots, P_n(\bar{x}), A_i < c_i, A_j > c_j, A_k \neq c_k, \ldots)$ and can be rewritten as

$$\forall \bar{x}(\neg P_1(\bar{x}) \vee \ldots \vee \neg P_n(\bar{x}) \vee A_i \geq c_i \vee A_j \leq c_j \vee A_k = c_k \vee \ldots) \quad (2)$$

This formula shows that since the repairs are done by attributes updates, the only way we have of solving an inconsistency is by fixing at least one of the values of a flexible attribute. This would imply to change the value of a flexible attribute $A_i$ to something equal or greater than $c_i$, to change the value of a flexible attribute $A_j$ to a value equal or smaller than $c_j$ or to change the value of attribute $A_k$ to $c_k$.

If $D$ is consistent wrt $IC$ then there is a unique fix $D' = D$ and all the values are the same as the original ones and therefore the proposition holds. If $D$ is inconsistent wrt $IC$ then there exists a tuple $t$ with at least one flexible attribute and a set $IC_t \subseteq IC$ such that for every $ic \in IC_t$ it holds $\mathcal{I}(D, ic, t) \neq \emptyset$. If $IC_t$ is an inconsistent set of constraints then there exists no local fix and the proposition holds. If $IC_t$ is consistent but there is at least one constraint with no flexible attributes involved then, since it is not possible to modify any attribute in order to satisfy the constraint, there is no local fix and the proposition holds.

So we are only missing to prove the proposition for $IC_t$ consistent and with at least one flexible attributes for each $ic$ in $IC_t$. From Lemma A.1 we know that there exists a local fix for $t$. Also, since $IC_t$ is consistent, using the same arguments as in proof of Lemma A.1, it is possible to define for each flexible attribute $A$ an interval such that if the value of $A$ is in it we would restore the consistency of the violation sets for constraints in $IC_t$ involving $t$. Then, we need to prove that if a value of an attribute, say $A$, of a local fix $t'$ of $t$ is different than the one in $t$, then the value corresponds to one of the closed limits of the interval for $A$. Let us assume that an attribute $A$ is restricted by the constraints to an interval $[c_l, c_u]$ and that the local fix $t'$ takes for attribute $A$ a value strictly smaller than $c_u$ and strictly greater than $c_l$. Without lost of generality we will assume that the value of attribute $A$ in $t$ is bigger than $c_u$. Let $t''$ be a tuple with the same values as $t'$ except that the attribute $A$ is set to $c_u$. $t''$ will have the same values in the hard attributes as $t$ and also $S(t, t') = S(t, t'')$ since the value of $A$ in $t''$ is still in the interval. We also have that $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$. This implies that $t'$ is not a local fix and we have reached a contradiction.

For the second part of the proposition, the proof of the first part can be easily extended to prove that the values in $D'$ will correspond to a border of a constraint in $IC$, because the LS-fixes are combination of local fixes. $\Box$

**Proof of Theorem 1:** Hilbert's 10th problem on existence of integer solutions to diophantine equations can be reduced to our problem. Given a diophantine equation, it is possible to construct a database $D$ and a set of ICs $IC$ such that the existence of a fix for $D$ wrt $IC$ implies the existence of a solution to the equation, and viceversa. $\Box$

**Proof of Lemma 1:** First for the skeptical semantics. Given a database instance $D$, consider the instance $(D, no)$ for $Cqa(\mathit{False}, IC, Sk)$, corresponding to the question "Is there a fix of $D$ wrt $IC$ that does not satisfy *False*?" has answer *Yes* iff the class of fixes of $D$ is empty. For the majority semantics, for the instance $(D, no)$ for $Cqa(\mathit{True}, IC, Maj)$, corresponding to the question "Is it not the case that the majority of the fixes satisfy *True*?", we get answer *yes* iff the set of fixes is empty. $\Box$

**Proof of Corollary 1:** From Theorem 1 and Lemma 1. $\Box$

**Proof of Lemma 2:** Let $\rho$ be the square distance between $D$ and $D'$ in Definition 1. The circle of radius $\rho$ around $D$ intersects the non empty "consistent" region that contains the database instances with the same schema and key values as $D$ and satisfy $IC$. Since the circle has a finite number of instances, the distance takes a minimum in the consistent region. $\Box$

**Proof of Proposition 1:** First of all, we notice that a linear denial with implicit equalities, i.e. occurrences of a same variable in two different database atoms, e.g. $\forall X, Y, Z \neg(R(X, Y), Q(Y, Z), Z > 3)$, can be replaced by its *explicit version* with explicit equalities, e.g. $\forall X, Y, Z, W \neg(R(X, Y), Q(W, Z), Y = W, Z > 3)$.

Let $n$ be the number of tuples in the database, and $l$ be the number of attributes which participate in $IC$. They are those that appear in built-in predicates in the explicit versions of the ICs that do not belong to a key or are equal to a key (because they are not allowed to change). For example, given the denial $\neg(P(X,Y), Q(X,Z), Y > 2)$, since its explicit version is $\neg(P(X,Y), Q(W,Z), Y > 2, X = W)$, the number $l$ is 1 (for $Y$) if $X$ is a key for $P$ or $Q$, and 3 if $X$ is not a key (for $Y, X, W$).

If there exist a fix $D'$ with $\Delta(D, D') \leq k$, then no value in a flexible attribute in $D'$ differs from its corresponding value (through the key value) in $D$ by more than $\sqrt{k}$. In consequence, the size of a fix may not differ from the original instance by more than $l \times n \times \mathit{bin}(k)/2$, where $\mathit{bin}(k)$ is the size of the binary representation of $k$. Thus, the size of a fix is polynomially bounded by the size of $D$ and $k$. Since we can determine in polynomial time if $D'$ satisfies the ICs and if the distance is smaller than $k$, we obtain the result. □

**Proof of Theorem 2:** Membership: According to Lemma 2, there is an LS-fix at a square distance $\leq k$ iff there is an instance $D'$ with the same key values that satisfies $IC$ at a square distance $\leq k$. We use Proposition 1.

Hardness: We can reduce Vertex Cover (VC) to $DFP(IC_0)$ for a fixed set of denials $IC_0$. Given an instance $(\mathcal{V}, \mathcal{E})$, $k$ for VC, consider a database $D$ with a relation $E(X,Y)$ and key $\{X,Y\}$ for the edges of the graph, and a relation for the vertices $V(X, \mathit{Chosen})$, where $X$ is the key and attribute *Chosen*, the only flexible attribute, is initially set to 0. The constraint $IC$: $\forall X, Y, C_1, C_2 \neg(E(X,Y) \wedge V(X,C_1) \wedge V(X,C_2) \wedge C_1 < 1 \wedge C_2 < 1)$ expresses that for any edge, at least one of the incident vertices is be covered. A vertex cover of size $k$ exists iff there exists an LS-fix of $D$ wrt $IC$ at a distance $\leq k$. The encoding is polynomial in the size of the original graph. □

**Proof of Theorem 3:** For hardness, linear denials are good enough. We reduce the graph 3-colorability problem to $NE(IC_0)$, for a fixed set $IC_0$ of ICs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with set of vertexes $\mathcal{V}$ and set of edges $\mathcal{E}$. Consider the following database schema, instance $D$, and set $IC_0$ of ICs:

1. Relation $Vertex(Id, Red, Green, Blue)$ with key $Id$ and domain $\mathbb{N}$ for the last three attributes, actually the only three flexible attributes in the database; they can be subject to changes. For each $v \in \mathcal{V}$ we have the tuple $(v, 0, 0, 0)$ in *Vertex* (and nothing else).

2. Relation $Edge(id_1, id_2)$; and for each $e = (v_1, v_2) \in \mathcal{E}$, there are the tuples $(v_1, v_2), (v_2, v_1)$ in *Edge*. This relation is not subject to any fix.

3. Relation $Tester(Red, Green, Blue)$, with extension $(1,0,0), (0,1,0), (0,0,1)$. This relation is not subject to any fix.

4. Integrity constraints:
$\forall ixyz \neg(Vertex(i,x,y,z), x < 1, y < 1, z < 1)$; $\forall ixyz \neg(Vertex(i,x,y,z), x > 1)$ (the same for $y, z$); $\forall ixyz \neg(Vertex(i,x,y,z), x = 1, y = 1, z = 1)$; $\forall ixyz \neg(Vertex(i,x,y,z), x = 1, y = 1)$; etc. $\forall ijxyz \neg(Vertex(i,x,y,z), Vertex(j,x,y,z), Edge(i,j), Tester(x,y,z)$.

The graph is 3-colorable iff the database has a fix wrt $IC_0$. The reduction is polynomial in the size of the graph. If there is an LS-fix of the generated instance, then the graph is 3-colorable. If the graph is colorable, then there is a consistent instance with the same key values as the original instance; then, by Lemma 2, there is an LS-fix.

For membership, it suffices to prove that if an LS-fix exists, then its square distance to $D$ is polynomially bounded by the size of $D$, considering both the number of tuples and the values taken by the flexible attributes.

We will show that if a fix $D'$ exists, then all the values in its flexible attributes are bounded above by the maximum of $n_1 + n + 1$ and $n_2 + n + 1$, where $n$ is the number of tuples in the database, $n_1$ is the maximum absolute value in a flexible attribute in $D$, and $n_2$ is the maximum absolute value of a constant appearing in the ICs.

The set of denial ICs put in disjunctive form gives us a representation for all the ways we have to restore the consistency of the database. So, we have a constraint of the form $\varphi_1 \wedge \varphi_2 \cdots \varphi_m$, where each $\varphi_i$ is a disjunction of negated database atoms and inequalities, e.g. something like $\neg P(X,Y,Z) \vee \neg R(X_1,Y_1) \vee X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$. Since fixes can be obtained by changing values of non key attributes, each tuple in a fix is determined by a set of constraints, each of which is a disjunction of atoms of the form $X_i \theta_i c_m$ or $X_i \neq Y_j$, where $\theta_i$ is an inequality of the form $\leq, \geq, <, >$. E.g. from $\neg P(X,Y,Z) \vee \neg R(X_1,Y_1) \vee X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$ we get $X \leq c_1 \vee Y \leq c_2 \vee Z \neq Y_1$, which for a specific tuple becomes $Y \leq c_2 \vee Z \neq Y_1$ if $X$ is part of the key and its specific value for the tuple at hand does not satisfy $X \leq c_1$ (otherwise we drop the constraint for that tuple). In any case, every tuple in a fix can take values in a space $S$ that is the intersection of the half-spaces defined by inequalities of the form $X_i \theta_i c_m$ minus the set of points determined by the non-equalities $X_i \neq Y_j$.

If there is a set of values that satisfies the resulting constraints, i.e. if there is an instance with the same key values that satisfies the ICs, then we can find an LS-fix at the right distance: if the difference between any value and $max(c_1, \cdots, c_l)$ is more than $n+1$ (the most we need to be sure the inequalities $X_i \neq Y_j$ are satisfied), then we systematically change values by 1, making them closer to the borders of the half-spaces, but still keeping the points within $S$. $\Box$

**Proof of Corollary 2:** *coNP*-hardness follows from Lemma 1 and Theorem 3. $\Box$

**Proof of Theorem 4:** We reduce 3-SAT's complement to LS-fix checking for a fixed schema and set of denials *IC*. We have a table $Lit(l, \bar{l})$ storing complementary literals (only), e.g. $(p, \neg p)$ if $p$ is one of the variables in the instance for SAT. Also a table $Cl$ storing tuples of the form $(\varphi, l, k)$, where $\varphi$ is a clause (we assume all the clauses have exactly 3 literals, which can be simulated by adding extra literals with unchangeable value 0 if necessary), $l$ is a literal in the clause, and $k$ takes value 0 or 1 (the truth value of $l$ in $\varphi$). The first two arguments are the key of $C$. Finally, we have a table $Aux(K,N)$, with key $K$ and flexible numerical attribute $N$, and a table $Num(N)$ with a hard numerical attribute $N$.

Given an instance $\Phi = \varphi_1 \wedge \cdots \wedge \varphi_m$ for 3-SAT, we produce an initial extension $D$ for the relations in the obvious manner, assigning arbitrary truth values to the literals, but making sure that the same literal takes the same truth value in every clause, and complementary literals take complementary truth values. $Aux$ contains $(0,0)$ as its only tuple; and $Num$ contains $(s+1)$, where $s$ is the number of different propositional variables in $\Phi$.

Consider now the following set of denials:
(a) $\neg(Cl(\varphi, L, U), U > 1)$; $\neg(Cl(\varphi, L, U), U < 0)$ (possible truth values).
(b) $\neg(Cl(\varphi, L, U),\ Cl(\psi, L,\ V),\ U \neq V,\ Aux(K,N),\ N \neq 0, N \neq Z)$ (same value for a literal everywhere).
(c) $\neg(Cl(\varphi, L, U),\ Cl(\psi, L', V), Lit(L, L'), U = V,\ Aux(K,N), N \neq\ 0, N \neq Z)$ (complementary literals).
(d) $\neg(Cl(\varphi, L, U), Cl(\varphi, L', V), Cl(\varphi, L'', W), U = V = W = 0, L \neq L', ..., Aux(K,N), N = 0$ (each clause becomes true).
(e) $\neg Num(Z), Aux(K,N), N \neq 0, N \neq Z)$ (possible values).

It holds that the formula is unsatisfiable iff the instance $D'$ that coincides with $D$ except for $Aux$ that now has the only tuple $(0, s+1)$ is an LS-fix of $D$ wrt *IC*. Thus, checking $D'$ for LS-fix is enough to check unsatisfiability.

For membership to *coNP*, for an initial instance $D$, instances $D'$ in the complement of $Fix(IC)$ have witnesses $D''$ that can be checked in polynomial time, namely instances $D''$ that have the same key values as $D$, satisfy the ICs, but $\Delta(D, D'') < \Delta(D, D')$. $\Box$

**Proof of Proposition 2:** The reduction can be established with a fixed set $IC_0$ of ICs. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consider a database with relations $V(X, Z), E(U, W)$, where $X$ is a key and $Z$ is the only flexible attribute and takes values in $\{0, 1\}$ (which can be enforced by means of the linear denials $\forall X \forall Z \neg(V(X, Z), Z > 1)$, $\forall X \forall Z \neg(V(X, Z), Z < 0)$ in $IC_0$). Intuitively, $Z$ indicates with 1 if the vertex $X$ is in the cover, and with 0 otherwise. Attributes $U, V$ are vertices and then, non numerical.

In the original database $D$ we have the tuples $V(e, 0)$, with $e \in \mathcal{V}$; and also the tuples $E(e_1,\ e_2)$ for $(e_1,\ e_2) \in \mathcal{E}$. Given the linear constraint $\forall X_1 Z_1 X_2 Z_2 \neg(V(X_1, Z_1), V(X_2, Z_2), E(X_1, X_2), Z_1 = 0, Z_2 = 0)$ in $IC_0$, the LS-fixes of the database are in one-to-one correspondence with the vertex covers of minimal cardinality.

For the query $Q^{(k)} : q(sum(Z)), sum(Z) < k$, with $q(sum(Z)) \leftarrow V(X, Z)$, the instance $(D, yes)$ for consistent query answering under brave semantics has answer *No*, (i.e. $Q^{(k)}$ is false in all fixes) only for every $k$ smaller than the minimum cardinality $c$ of a vertex cover. $\square$

**Proof of Corollary 3:** Let *IC* and a query $Q$ be given. The complement of CQA is in $NP^{coNP}$: Given an instance $D$, non deterministically choose an instance $D'$ with $D' \not\models Q$ and $D'$ a fix of $D$. The latter test can be done in *coNP* (by Theorem 4). But $NP^{coNP} = NP^{\Sigma_1^P} = \Sigma_2^P$. In consequence, CQA belongs to $co\Sigma_2^P = \Pi_2^P$. $\square$

**Proof of Proposition 3:** By reduction from the *MAXSNP*-hard problem *B-Minimum Vertex Cover* (BMVC), which asks to find a minimum vertex cover in a graph whose nodes have a bounded degree [13, chap. 10]. We start by encoding the graph as in the proof of Proposition 2. We also use the same initial database $D$. Every LS-fix $D'$ of $D$ corresponds to a minimum vertex cover $\mathcal{V}'$ for $\mathcal{G}$ and vice versa, and it holds $|\mathcal{V}'| = \Delta(D, D')$. This gives us an $L$-reduction from BMVC to *DFP* [17]. $\square$

**Proposition A.2.** For the class of local denials, *DFP* is *NP*-complete. $\square$
**Proof:** Membership follows from Theorem 2. For hardness, we can do the same reduction as in Theorem 2, because the ICs used there are local denials. $\square$

**Lemma A.2.** Given a database $D$ and a set of consistent linear denials *IC* with joins only between hard attributes and with at least one flexible attribute in each of them, there will always exist an LS-fix $D'$ of $D$ wrt *IC*. $\square$
**Proof:** As shown in proof of Lemma A.1 for every flexible attribute in $F$ it is possible to define, using the integrity constraints in *IC*, an interval $[c_l, c_u]$ such that if the value of attribute $A$ is in that interval there is no constraint $ic \in IC$ with a built-in involving $A$ such that $\mathcal{I}(D, ic, t) \neq \emptyset$. Let $D''$ be a database constructed in the following way: for every tuple $t \in D$ such that the value of a flexible attribute does not belong to its interval, replace its value by any value in the interval. Clearly $D'$ will have the same schema and key values and will satisfy *IC* but will not necessarily differ from the original database in a minimal way. By Lemma 2 we know there exists an LS-fix $D'$ for $D$ wrt *IC*. $\square$

**Definition 8.** Given a database $D$ and a set of ICs *IC*, a local fix $t'$ for a tuple $t$ *does not generate new violations* if $\bigcup_{ic \in IC}(\bigcup_{l \in D'} \mathcal{I}(D', ic, l) \smallsetminus \bigcup_{l \in D} \mathcal{I}(D, ic, l)) = \emptyset$ for $D' = (D \smallsetminus \{t\}) \cup \{t'\}$. $\square$

**Lemma A.3.** For a set *IC* of local denials, if $t'$ is a local fix of a tuple $t$, then $t'$ does not generate new violations[3] in database $D$ wrt *IC*. Furthermore, this holds also for $t'$ a "relaxed" local fix where the distance to $t$ is not necessarily minimal $\square$

[3]c.f. Definition 8

**Proof:** Tuple $t'$ can only differ from $t$ in the value of flexible attributes. Let us assume that one of the modified values was for an attribute $A$. Since we have local constraints, attribute $A$ can only be in the constraints related either to $<$ and $\leq$ or to $>$ and $\geq$, but not both. Without lost of generality, we will assume that the constraint is written as in equation 1 and that $A$ is related only to $>$ and $\geq$. Since $t'$ is a fix, $S(t,t')$ is not empty and there is a set $IC_t$ of constraints for which $t'$ solves the inconsistency in which $t'$ was involved. There is an interval $[c_l, +\infty)$ for $A$ that can be obtained by the limits given in $IC_t$ that show the values of $A$ that would force the satisfaction of the constraints in $IC_t$ that have attribute $A$ in an inequality. This shows that the value of attribute $A$ in $t'$ is bigger than the value of $A$ in $t$.

For $D' = (D \smallsetminus \{t\}) \cup \{t'\}$ we need to prove that $\bigcup_{ic \in IC}(\bigcup_{l \in D'} \mathcal{I}(D', ic, l) \smallsetminus \bigcup_{l \in D} \mathcal{I}(D, ic, l)) = \emptyset$. By contradiction let us assume that for a constraint $ic \in IC$ there exists a violation set $I$ such that $I \in \bigcup_{l \in D'} \mathcal{I}(D', ic, l)$ and $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$. There are two cases to consider:

- $(I, ic) \in S(t,t')$. Then $I \in \mathcal{I}(D, ic, t)$, but since we wanted an $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$ this is not possible.
- $(I, ic) \notin S(t,t')$. Then we have two possibilities $I \notin \mathcal{I}(D, ic, t)$ or $((I \smallsetminus \{t\}) \cup \{t'\}) \not\models ic$.
  - Let us consider first that $I \notin \mathcal{I}(D, ic, t)$. We have that $I \in \bigcup_{l \in D'} \mathcal{I}(D', ic, l)$ and since $t'$ is the only difference between $D$ and $D'$ we have $I \in \mathcal{I}(D', ic, t')$. Since all the constraints can only have attribute $A$ with $>$ or $\geq$ we now that in particular $ic$ does. Since $I \notin \mathcal{I}(D, ic, t)$ we know that $A$ satisfied the condition in $ic$ and since we know that $t'$ has a bigger value than in $t$, it is not possible to generate an inconsistency in $D'$. We have reached a contradiction.
  - Let us consider $((I \smallsetminus \{t\}) \cup \{t'\}) \not\models ic$. Then $I \in \mathcal{I}(D', ic, t')$. From our assumption $I \notin \bigcup_{l \in D} \mathcal{I}(D, ic, l)$.This is the same situation analyzed in previous item.

In all the cases we have reached contradiction and therefore the proposition is proved. Since we never used the property of minimal distance between $t'$ and $t$, the second part of the Lemma is also proved. $\square$

**Proposition A.3.** For local denials it always exists an LS-fix for a database $D$; and for every LS-fix $D'$, $D' \smallsetminus D$ is a set of local fixes. Furthermore, for each violation set $(I, ic)$, there is a tuple $t \in I$ and a local fix $t'$ for $t$, such that $(I, ic) \in S(t,t')$. $\square$

**Proof:** Since each attribute A can only be associated to $<$ or $>$ built-ins, but not both, it is clear that set of local denials is always consistent. By Lemma A.2, there always exists a fix $D'$. Now we need to prove that $D' \smallsetminus D$ is a set of local fixes. By contradiction assume that $t' \in (D' \smallsetminus D)$ is not a local fix of the tuple $t$. This can happen in the following situations:

- $t$ was consistent. From Lemma A.3 we know that no new inconsistencies can be added by the modifications done to the other tuples and therefore $t$ is not related to any inconsistency. Then $D^\star = D' \smallsetminus \{t'\} \cup \{t\}$ is also consistent and $\Delta(D, D^\star) < \Delta(D, D')$. But $D'$ is an LS-fix so this is not possible.
- $t$ is involved at least in one violation set. If $S(t,t') = \emptyset$ then $t'$ is not solving any violation set and therefore $D^\star = D' \smallsetminus \{t'\} \cup \{t\}$ is also consistent and $\Delta(D, D^\star) < \Delta(D, D')$. But $D'$ is an LS-fix so this is not possible. Now, if $S(t,t') \neq \emptyset$, from Lemma A.2, considering $D = \{t\}$ and $IC = \{ic | (I, ic) \in S(t,t')\}$, there exists an LS-fix $D'$ of $D$, i.e. there exists a local fix $t''$ such that $S(t,t'') = S(t,t')$. Since $t''$ is a local fix we know that $\Delta(\{t\}, \{t''\}) \leq \Delta(\{t\}, \{t'\})$. They cannot be equal that would imply that $t'$ is a local fix and it is not. Then $D^\star = D' \smallsetminus \{t'\} \cup \{t\}$ is also consistent and $\Delta(D, D^\star) < \Delta(D, D')$. Again, this is not possible because $D'$ is an LS-fix

The second part of the proposition can be proved using Lemma A.2 and considering a database $D = I$ and a set of constraints $IC = \{ic\}$. $\square$

**Proposition A.4.** For a database $D$ and a set of local denial constraints $IC$:

1. For a set of local fixes $\{t_1, \dots t_n\}$ of a tuple $t$ there always exists a local fix $t^\star$ such that $S(t, t^\star) = \bigcup_{i=1}^n S(t, t_i)$.
2. For local fixes $t'$, $t''$ and $t'''$ of a tuple $t$ with $S(t, t''') = S(t, t') \cup S(t, t'')$, it holds that $\Delta(\{t\}, \{t'''\}) \leq \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. □

**Proof:** First we will prove item (1). Let $IC_t = \{ic | \mathcal{I}(D, ic, t) \neq \emptyset$ and $IC_S(t, t') = \{ic | (I, ic) \in S(t, t')\}$. ¿From Lemma A.2, considering $D = \{t\}$ and $IC$ any subset of $IC_t$, there always exists an LS-fix $D'$ of $D$. This LS-fix is a local fix of tuple $t$ with $IC_S(t, t') = IC$. Since we can find a local fix for any $IC \subseteq S_t$ then clearly the lemma can be satisfied.

Now we will prove item (2). If the flexible attributes that where modified in $t'$ and $t''$ are disjoint, then $t'''$ when combining the modifications I'll get $\Delta(\{t\}, \{t'''\}) = \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Now, we will consider the case were $t'$ and $t''$ have at least one flexible attribute, say $A$ that is modified by both local fixes. In this case $t'''$ will have a value in $A$ that solves the inconsistencies solved by and $t'$ and $t''$. This value will in fact correspond to the value of $A$ in $t'$ or $t''$ and therefore we will have that $\Delta(\{t\}, \{t'''\}) < \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Let $M$ be the set of attributes that are modified both by $t'$ and $t''$, we can express the relation as follows: $\Delta(\{t\}, \{t'''\}) = \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t'''))^2 = \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t'))^2 + \sum_{A \in \mathcal{F}} (\pi_A(t) - \pi_A(t''))^2 - \sum_{A \in \mathcal{M}} Min\{(\pi_A(t) - \pi_A(t'))^2, (\pi_A(t) - \pi_A(t''))^2\}$ □

**Proposition A.5.** If an optimal cover $\mathcal{C}$ for the instance $(U, \mathcal{S})$ of $MWSCP$ has more than one $S(t, t')$ for a tuple $t$, then there exists another optimal cover $\mathcal{C}'$ for $(U, \mathcal{S})$ with the same total weight as $\mathcal{C}$ but with one $t'$ such that $S(t, t') \in \mathcal{C}$. Furthermore, $D(\mathcal{C})$ is an LS-fix of $D$ wrt $IC$ with $\Delta(D, D(\mathcal{C}))$ equal to the total weight of the cover $\mathcal{C}$. □

**Proof:** First we will prove the first part of the Proposition. Let us assume that we have $\{S(t, t'), S(t, t'') \in \mathcal{C}\}$. From Proposition A.4 there exists an $S(t, t''') \in \mathcal{S}$ such that $S(t, t''') = S(t, t') \cup S(t, t'')$, i.e such that it covers the same elements as $S(t, t')$ and $S(t, t'')$. From Proposition A.4 $\Delta(\{t\}, \{t'''\}) \leq \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$ and therefore that weight of $S(t, t''')$ is smaller or equal than the sum of the weight of the original two sets. If $\Delta(\{t\}, \{t'''\}) < \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$ we would have that $\mathcal{C}$ is not an optimal solution so this is not possible. Then $\Delta(\{t\}, \{t'''\}) = \Delta(\{t\}, \{t'\}) + \Delta(\{t\}, \{t''\})$. Then, if we define $\mathcal{C}' = (\mathcal{C} \smallsetminus \{S(t, t'), S(t, t'')\}) \cup \{S(t, t''')\}$ we will cover all the elements and we will have the same optimal weight.

Now we need to prove that given $D(\mathcal{C})$ is an LS-fix. $D(\mathcal{C})$ is obtained by first calculating $\mathcal{C}'$ and therefore we have an optimal cover with at most one $S(t, t')$ for each tuple $t$. Then $D(\mathcal{C})$ is obtained by replacing $t$ by $t'$ for each $S(t, t') \in \mathcal{C}$. It is direct that $D(\mathcal{C})$ has the same schema as $D$ and that it satisfies the key constraints. Now, since $\mathcal{C}'$ covers all the elements, all the inconsistencies in $D$ are solved in $D(\mathcal{C})$. From Lemma A.3 the local fixes $t'$ do not add new violations and therefore $D(\mathcal{C}) \models IC$. We are only missing to prove that $D(\mathcal{C})$ minimizes the distance from $D$. Clearly $\Delta(D, D(\mathcal{C})) = \sum_{t \in D} \Delta(\{t\}, \{t'\}) = \sum_{S(t,t') \in \mathcal{C}'} w_{S(t,t')} = \sum_{S(t,t') \in \mathcal{C}} w_{S(t,t')} = w$. So, since the optimal solution minimizes $w$, $\Delta(D, D(\mathcal{C}))$ is minimum and $D(\mathcal{C})$ is an LS-fix □

**Proposition A.6.** For every LS-fix $D'$ of a database $D$ wrt a set of local denials $IC$, there exists an optimal cover $\mathcal{C}$ for an instance $(U, \mathcal{S})$ of $MWSCP$ such that $D' = D(\mathcal{C})$. □

**Proof:** To prove it it is enough to construct this optimal cover. Let $\mathcal{C} = \{S(t, t') | t' \in (D' \smallsetminus D)$. By definition $\mathcal{C}' = \mathcal{C}$ and $D(\mathcal{C}) = D'$. We need to prove that $\mathcal{C}$ is an optimal cover. Since $D'$ is consistent, all the violation sets were solved and therefore $\mathcal{C}$ is a cover. Also, since $\Delta(D, D') = \Delta(D, D(\mathcal{C})) = w$ and $\Delta(D, D')$ is minimum, $\mathcal{C}$ minimizes the weight and therefore is an optimal cover. □

**Proposition A.7.** The transformation from *DFOP* to *MWSCP*, and the construction of the instance $D(\mathcal{C})$ from a set cover $\mathcal{C}$ can be done in polynomial time in the size of $\mathcal{D}$.

**Proof:** We have to establish that the transformation of *DFOP* into *MWSCP* given above is an $L$-reduction [17]. So, it remains to verify that the reduction can be done in polynomial time in the size of instance $D$ for $DFP(IC)$, i.e. that $\mathcal{G}$ can be computed in polynomial time in $n$, the number of tuples in $D$. Notice that if $m_i$ the number of database atoms in $ic_i \in IC$, and $m$ the maximum value of $m_i$ there are at most $n^{m_i}$ hyper-edges associated to $ic_i \in IC$, each of them having between 1 to $m$ tuples. We can check that the number of sets $S(t,t')$ and their weights are polynomially bounded by the size of $D$. There is one $S(t,t')$ for each local fix. Each tuple may have no more than $|\mathcal{F}| \times |IC|$ local fixes, where $\mathcal{F}$ is the set of flexible attributes.

The weight of each $S(t,t')$ is polynomially bounded by the maximum absolute value in an attribute in the database and the maximum absolute value of a constant appearing in $IC$ (by an argument similar to the one given in the proof of Proposition 1).

With respect to $D(\mathcal{C})$, the number of sets in $\mathcal{S}$ is polynomially bounded by the size of $D$, and since $\mathcal{C} \subseteq \mathcal{S}$, $\mathcal{C}$ is also polynomially bounded by the size of $D$. To generate $\mathcal{C}'$ it is necessary to search through $\mathcal{S}$. Finally, in order to replace $t$ in $D$ for each tuple $t'$ such that $S(t,t') \in \mathcal{C}$ we need to search through $D$. □

**Proposition A.8.** Given a cover $\hat{\mathcal{C}}$ obtained with the approximation algorithm for an instance $(U, \mathcal{S})$ of *MWSCP*, $D(\hat{\mathcal{C}})$ will have the same schema as $D$, will satisfy the key and the set of constraints $IC$, but will not necessarily minimize the distance to the original database $D$. We also have that $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w}$, where $\hat{w}$ is the total weight of $\hat{\mathcal{C}}$, and therefore that $\Delta(D, D(\hat{\mathcal{C}}))$ is a better approximation than $\hat{w}$ to the optimal solution □

**Proof:** Using the same arguments as in the proof of Proposition A.5 we have that since $\hat{\mathcal{C}}$ is a cover then $D(\hat{\mathcal{C}})$ has the same schema as $D$, satisfies the keys and satisfies $IC$. What we are not sure of is if $D(\hat{\mathcal{C}})$ minimizes the distance to $D$. Before we had that $\hat{\mathcal{C}}'$ had the same weight as $\mathcal{C}$. Now, from Proposition A.4 $\Delta(\{t\},\{t'''\}) \leq \Delta(\{t\},\{t'\}) + \Delta(\{t\},\{t''\})$, and since $\hat{\mathcal{C}}$ is not optimal they are not necessarily equal so, $w_{\hat{\mathcal{C}}'} <= \hat{w}$. Then $\Delta(D, D(\hat{\mathcal{C}})) \leq \hat{w}$ but not necessarily minimal. □

**Proof of Theorem 5:** Based on the tractability results in [10], it suffices to show that the LS-fixes for a database $D$ are in one-to-one and polynomial time correspondence with the repairs using tuple deletions [2, 7] for a database $D'$ wrt a set of key dependencies.

Since we have 1DADs, the violation sets will have a single element, then, for an inconsistent tuple $t$ wrt a constraint $ic \in IC$, it holds $\mathcal{I}(D, ic, t) = \{t\}$. Since all the violation sets are independent, in order to compute a fix for $D$, we have to generate independently all the local fixes $t'$ for all inconsistent tuples $t$ such that $(\{t\}, ic) \in S(t,t')$, with $ic \in IC$; and then combine them in all possible ways.

Those local fixes can be found by considering all the candidate fixes (not necessarily LS-minimal) that can obtained by combining all the possible limits for each attribute provided by the ICs (c.f. Proposition A.1); and then checking which of them satisfy $IC$, and finally choosing those that minimize $\Delta(\{t\},\{t'\})$. There are at most $2^{|\mathcal{F}|}$ possible candidate fixes, where $\mathcal{F}$ is the set of flexible attributes.

Let us now define a database $D'$ consisting of the consistent tuples in $D$ together with all the local fixes of the inconsistent tuples. By construction, $D$ and $D'$ share the same keys. Since each inconsistent tuple in $D$ may have more than one local fix, $D'$ may become inconsistent wrt its key constraints. Each repair for $D'$, obtained by tuple deletions, will choose one local fix for each inconsistent tuple $t$ of $D$, and therefore will determine an LS-fix of $D$ wrt $IC$. □

**Proof of Proposition 4:** The *NP*-complete *PARTITION* problem [11] can be reduced to this case for a fixed set of 1DADs. Let a $A$ be a finite set, whose elements $a$ have integer sizes $s(a)$. We need to determine if there exists a subset $S$ of $A$, such that $\sum_{a \in S} s(a) = n := (\sum_{a \in A} s(a))/2$.

We use two tables: *Set*(*Element*, *Weight*), with key {*Element*, *Weight*}, containing the tuples $(a, s(a))$ for $a \in A$; and $\mathit{Selection}(\mathit{Element}, X, Y)$, with key *Element*, flexible numerical attributes $X, Y$ (the partition of $A$) taking values 0 or 1 (which can be specified with 1DADs), and initially containing the tuples $(a, 0, 0)$ for $a \in A$. Finally, we have the 1DAD $\forall E, X, Y \neg(\mathit{Selection}(E, X, Y), X < 1, Y < 1)$.

There is a one-to-one correspondence between LS-repairs of the original database and partitions $X, Y$ of $A$ (collecting the elements with value 1 in either $X$ or $Y$). Then, there is a partition with the desired property iff the query $Q: \ (\mathit{Set}(E, W), \mathit{Selection}(E, X, Y), X = 1, \mathit{sum}(W) = n)$ has answer *yes* under the brave semantics. The query used in this proof is acyclic and belongs to the class $\mathcal{C}_{\mathit{Tree}}$. □

**Proof of Proposition 5:** By reduction from a variation of *Independent Set*, for graphs whose vertices have all the same degree. It remains *NP*-hard as a special case of *Independence Set for Cubic Planar Graphs* [12]. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with degree $d$, and a minimum bound $k$, we create a relation $\mathit{Vertex}(V, C_1, C_2)$, where the key $V$ is a vertex and $C_1, C_2$ are flexible and taking values 0 or 1, but all equal to 0 in the initial instance $D$. It is subject to the denial $\mathit{IC}: \forall V, C_1, C_2 \neg(\mathit{Vertex}(V, C_1, C_2), C_1 < 1, C_2 < 1)$. A second relation $\mathit{Edge}(V_1, V_2, W)$, with hard attributes only, contains the tuples $(v_1, v_2, 1)$ for $(v_1, v_2) \in \mathcal{E}$ or $(v_2, v_1) \in \mathcal{E}$. Every vertex $v$ appears in each argument in exactly $d$ tuples.

Consider the ground query aggregate conjunctive query $Q$:
$q(\mathit{sum}(W)) \leftarrow \mathit{Edge}(V_1, V_2, W), \mathit{Vertex}(V_1, C_{11}, C_{12}), \mathit{Vertex}(V_2, C_{21}, C_{22}), C_{11} = 1, C_{21} = 0)$.

We are interested in the maximum value for $Q$ in $\mathit{Fix}(D, \mathit{IC})$, i.e. the *min-max answer* introduced in [3].
In a fix, the tuples $(v_1, 1, 0)$ and $(v_2, 0, 1)$ in *Vertex* partition $\mathcal{V}$, so in a fix, the tuples of the form $(v, 1, 0)$ determine a subset of $\mathcal{V}$, and every set of vertices can be obtained in this way. In particular, an independent set $I$ corresponds to the tuples of the form $\mathit{Vertex}(v, 1, 0)$ in a fix $D(I)$. In every such a fix $D(I)$, the answer to $Q$ will be $d \times |I|$, because the predicate $\mathit{Edge}(V_1, V_2, W)$ in the query will be satisfied one and only one time for every edge incident to vertex belonging to $I$.

Then, among independent set fixes, the maximum value for Q is $d \times m$, where $m$ is the maximum cardinality of an independent set. Wrt to answers to $Q$ from fixes that do not correspond to independent sets, we have two cases. If $S \subseteq \mathcal{V}$ is determined by the tuples $(v, 1, 0)$ is a fix $D'$, and $S \subsetneqq I$ for some independent set $I$, then $Q(S) < Q(D(I)) \leq d \times m$.

If $S$ is not contained in any independent set, it does contain an independent set $I$, $I \subseteq S$, that cannot be extended to a larger independent set still contained in $S$. For the maximal independent $I'$ that contains $I$ it holds $Q(S) < Q(D(I')) \leq d \times m$.

In consequence, the min-max answer for $Q$ is $d \times m$; and then there is an independent set of size at least $k$ iff *min–max answer to* $Q \geq k \times d$. □

## A.2 An Example for Theorem 1

Consider the diophantine equation

$$2x^3y^2 + 3xy + 105 = x^2y^3 + y^2. \tag{3}$$

Each term $t$ in it will be represented by a relation $R(t)$ with 8 attributes taking values in $\mathbb{N}$: three, $X_1, X_2, X_3$, for the maximum exponent of $x$, three, $Y_1, Y_2, Y_3$, for the maximum exponent of $y$, one, $C$, for the constant terms, plus a last one, $K$, for a key. Value 0 for a non-key attribute indicates that the term appears in $t$, otherwise it gets value 1. We introduce as many tuples in $R(t)$ as the coefficient of the term; they differ only in the key value. We will see that only the 0 values will be subject to fixes. These are the relations and their ICs:

| $R(2x^3y^2)$ | $X_1$ | $X_2$ | $X_3$ | $Y_1$ | $Y_2$ | $Y_3$ | $C$ | $K$ |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 |

For this table we have the following set, $IC(2x^3y^2)$, of ICs:
$\forall x_1 \cdots x_8 \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge x_1 \neq x_2)$, $\forall x_1 \cdots x_8 \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge x_2 \neq x_3)$,
$\forall x_1 \cdots x_8 \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge x_5 \neq x_6)$, $\forall x_1 \cdots x_8 \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge x_4 \neq 1)$,
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(2x^3y^2)(x_9,\cdots,x_{16}) \wedge x_1 \neq x_9)$
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(2x^3y^2)(x_9,\cdots,x_{16}) \wedge x_5 \neq x_{13})$.

| $R(3xy)$ | $X_1$ | $X_2$ | $X_3$ | $Y_1$ | $Y_2$ | $Y_3$ | $C$ | $K$ |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 3 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 4 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 5 |

$IC(3xy)$:
$\forall x_1 \cdots x_{16} \neg(R(3xy)(x_1,\ldots,x_8) \wedge R(3xy)(x_9,\ldots,x_{16}) \wedge x_3 \neq x_{11})$,
$\forall x_1 \cdots x_{16} \neg(R(3xy)(x_1,\ldots,x_8) \wedge R(3xy)(x_9,\cdots,x_{16}) \wedge x_6 \neq x_{14})$,
$\forall x_1 \cdots x_8 \neg(R(3xy)(x_1,\ldots,x_8) \wedge x_1 \neq 1)$, $\forall x_1 \cdots x_8 \neg(R(3xy)(x_1,\ldots,x_8) \wedge x_2 \neq 1)$,
$\forall x_1 \cdots x_8 \neg(R(3xy)(x_1,\ldots,x_8) \wedge x_4 \neq 1)$,
$\forall x_1 \cdots x_8 \neg(R(3xy)(x_1,\ldots,x_8) \wedge x_5 \neq 1)$.

| $R(105)$ | $X_1$ | $X_2$ | $X_3$ | $Y_1$ | $Y_2$ | $Y_3$ | $C$ | $K$ |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 | 1 | 105 | 6 |

$IC(105)$:
$\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_1 \neq 1)$, $\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_2 \neq 1)$,
$\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_3 \neq 1)$, $\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_4 \neq 1)$,
$\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_5 \neq 1)$, $\forall x_1 \cdots x_8 \neg(R(105)(x_1,\ldots,x_8) \wedge x_6 \neq 1)$,
$\forall x_1 \cdots x_6 \neg(105(x_1,\cdots,x_6) \wedge x_7 \neq 105)$.

Similar tables $R(x^2y^3)$ and $R(y^2)$ and corresponding sets of ICs are generated for the terms on the RHS of (3).

Next we need ICs that are responsible for making equal all $x$s and $y$s in all terms of the equation:
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(3xy)(x_9,\cdots,x_{16}) \wedge x_1 \neq x_{11})$,
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(3xy)(x_9,\ldots,x_{16}) \wedge x_5 \neq x_{13})$
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(x^2y^3)(x_9,\cdots,x_{16}) \wedge x_1 \neq x_{10})$
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(x^2y^3)(x_9,\ldots,x_{16}) \wedge x_5 \neq x_{12})$
$\forall x_1 \cdots x_{16} \neg(R(2x^3y^2)(x_1,\ldots,x_8) \wedge R(y^2)(x_9,\ldots,x_{16}) \wedge x_5 \neq x_{13})$.

Now we construct a single table $R(equ)$ that represents equation (3) by appending the previous tables:

| $R(equ)$ | $X_1$ | $X_2$ | $X_3$ | $Y_1$ | $Y_2$ | $Y_3$ | $C$ | $K$ |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 2 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 3 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 4 |
| | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 5 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 105 | 6 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 7 |
| | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 8 |

We need ICs stating the correspondence between the terms in the tables $R(t)$ and table $R(equ)$:
$\forall x_1 \cdots x_{16} \neg(R(equ)(x_1, \ldots, x_8) \wedge R(2x^3y^2)(x_9, \ldots, x_{16}) \wedge x_8 = x_{16} \wedge x_1 \neq x_9)$,
$\forall x_1 \cdots x_{16} \neg(R(equ)(x_1, \ldots, x_8) \wedge R(2x^3y^2)(x_9, \ldots, x_{16}) \wedge x_8 = x_{16} \wedge x_2 \neq x_{10})$,
$\ldots \qquad \ldots \qquad \ldots$
$\forall x_1 \cdots x_{16} \neg(R(equ)(x_1, \ldots, x_6) \wedge R(y^2)(x_7 \cdots x_{16}) \wedge \ x_8 = x_{16} \wedge x_7 \neq x_{15})$.

Finally, we have one aggregate constraint that is responsible for making equal the LHS and RHS of equation (3):
$sum_{R(equ)}(x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \ : x_6 < 7) \ = \quad sum_{R(equ)}(x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \ : x_6 > 6)$.

If the database has a fix, then there is an integer solution to the diophantine equation. If the equation has a solution $s$, then there is an instance $R(equ)'$ corresponding to $s$ that satisfies the ICs. By Lemma 2, there is an LS-fix of the database.

The reduction could be done with the table $R(equ)$ alone, making all the ICs above to refer to this table, but the presentation would be harder to follow.