Available online at www.sciencedirect.com
ScienceDirect
Information Systems 33 (2008) 407–434
www.elsevier.com/locate/infosys

The complexity and approximation of fixing numerical attributes in databases under integrity constraints[☆]

Leopoldo Bertossi[a,*], Loreto Bravo[b], Enrico Franconi[c], Andrei Lopatenko[d]

[a]*Carleton University, School of Computer Science, Ottawa, Canada*
[b]*University of Edinburgh, School of Informatics, UK*
[c]*Free University of Bozen–Bolzano, Faculty of Computer Science, Italy*
[d]*Google Inc., Mountain View, CA, USA*

Abstract

Consistent query answering is the problem of characterizing and computing the semantically correct answers to queries from a database that may not satisfy certain integrity constraints. Consistent answers are characterized as those answers that are invariant under all minimally repaired versions of the original database. We study the problem of repairing databases with respect to denial constraints by fixing integer numerical values taken by attributes. We introduce a quantitative definition of database repair, and investigate the complexity of several decision and optimization problems. Among them, Database Repair Problem (*DRP*): deciding the existence of repairs within a given distance to the original instance, and *CQA*: deciding consistency of answers to simple and aggregate conjunctive queries under different semantics. We provide sharp complexity bounds, identifying relevant tractable and intractable cases. We also develop approximation algorithms for the latter. Among other results, we establish: (a) The Δ_2^P-hardness of *CQA*. (b) That *DRP* is *MAXSNP*-hard, but has a good approximation. (c) The intractability of *CQA* for aggregate queries for one database atom denials (plus built-ins), and also that it has a good approximation.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Integrity constraints; Inconsistent databases; Consistent query answering; Database repairs

1. Introduction

Integrity constraints (ICs) are used to impose semantics on a database. In this way, the database becomes an accurate model of an application domain. Database management systems or application programs enforce the satisfaction of the ICs by either rejecting undesirable updates or by executing additional compensating actions. However, there are many situations where we need to interact with a database that is inconsistent wrt certain desirable ICs. An important problem in database research consists in characterizing and retrieving consistent data from inconsistent databases, in particular, consistent answers to queries [1].

From the logical point of view, consistently answering a query on an inconsistent database amounts to evaluating the truth of a formula against a particular *class* of first-order relational

[☆]Dedicated to the memory of Alberto Mendelzon. Our research on this topic started with conversations with him. Alberto was always generous with his time, advice and ideas.

[*]Corresponding author.

E-mail addresses: bertossi@scs.carleton.ca (L. Bertossi), lbravo@inf.ed.ac.uk (L. Bravo), franconi@inf.unibz.it (E. Franconi), lopatenko@google.com (A. Lopatenko).

structures [2]. This process is quite different from usual truth or query evaluation on a single structure, namely the relational database at hand. In our case, the class under consideration is formed by alternative instances that are consistent, i.e. satisfy the ICs, and minimally differ from the original database, the so-called *repairs* of the latter. What is consistently true in the original instance corresponds to what is classically true of *all* repairs. Obviously, the notion of repair depends upon particular notions of difference between database instances and minimality.

Certain database applications, such as census, demographic, financial, and experimental data, contain quantitative data usually associated to nominal or qualitative data. For example, the number of children associated to a household identification code (or address); and the measurements associated to a sample identifier. Often this kind of data contains 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 is negative; or if the age of a mother is less than those of her offsprings. These restrictions can be expressed with denial ICs, which prevent attributes from taking certain combinations of values [3]. Other restrictions may be expressed with aggregation ICs. For example, the maximum concentration of certain toxin in a sample may not exceed a known threshold; or the number of married men and married women must be the same.

Inconsistencies in numerical data can be resolved by changing individual attribute values, while keeping values in the keys, e.g. without changing the household code, the number of children is decreased considering the admissible values. More precisely, we consider the problem of fixing integer numerical data wrt certain constraints while (a) keeping the attribute values in the keys of the relations, and (b) minimizing the quantitative global distance between the original and modified instances. Since the problem may admit several global solutions, each of them involving possibly many individual changes, we are interested in characterizing and computing data and properties that remain invariant under any of these repair processes. We concentrate on denial constraints; and conjunctive queries, with or without aggregation.

Database repairs have been 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] (cf. [1,4,5] for surveys). An answer to a query is consistent if 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. Changing the value of a particular attribute can be modelled as a deletion followed by an insertion, but this may not correspond to a minimum repair.

In certain applications it may make more sense to correct (update) values in certain numerical attributes only. This 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. We consider fixable attributes that take integer values. Only in these fixable attributes we allow for changes of values with the purpose of restoring consistency. In consequence, obtaining a repair becomes a numerical constraint satisfaction problem, where the constraints are given by the denials. The additional requirement in this problem is that the solutions, i.e. instances, should stay *close* to the initial instance.

Example 1. Consider a network traffic database D storing flow measurements of links in a network. This network has two types of links, labelled 0 and 1, with maximum flows 1000 and 1500, resp. The following database D is inconsistent wrt this constraint on the values that flows may take.

Traffic	*Time*	*Link*	*Type*	*Flow*
	1.1	a	0	1100
	1.1	b	1	900
	1.3	b	1	850

Under the tuple and set oriented semantics of repairs [2], there is a unique repair, namely deleting tuple *Traffic*$(1.1, a, 0, 1100)$. However, we have two options that may make more sense than deleting the flow measurement, namely updating the violating tuple to *Traffic*$(1.1, a, 0, 1000)$ or to *Traffic*$(1.1, a, 1, 1100)$. These alternatives would satisfy the implicit requirement that the numbers should not change too much.

In order to define a sensible distance function, for comparing alternative repairs to the original instance, we think that the numerical nature and the magnitude of these changes have to be considered. In this paper we start from the assumption that keeping the *overall and absolute variation* of values small and in balance is something desirable. A natural and usual way to achieve this goal consists in minimizing the square distance between the initial instance and a repair. For more flexibility, we allow for different weights to be assigned to the fixable attributes, and these weights are brought into the distance formula. In the same spirit, other distances between database instances, as an alternative to the Euclidean or L_2 distance that we investigate in this paper, could be considered (cf. Section 7.4). Specific repairs and approximations may be different under other distance functions, e.g. the "city distance" L_1 (the sum of absolute differences), but the general (in)tractability and approximation results remain.

The problem of attribute-based correction of census data forms is addressed in [3] using disjunctive logic programs with stable model semantics. Several underlying and implicit assumptions that are necessary for that approach to work are made explicit and used here, extending the semantic framework of [3]. However, in that work the numerical nature of some attributes is not brought into the model, and the distance just counts the number of changes, no matter how big or small they are.

Update-based repairs for restoring consistency are also studied in [6], where changing values in attributes in a tuple is made a primitive repair action. Semantic and computational problems around CQA are analyzed from this perspective. However, peculiarities of changing numerical attributes are not considered, and more importantly, the distance between databases instances used in [6,7] is based on set-theoretic homomorphisms, but is not quantitative, as in this paper.

We provide semantic foundations for repairs that are based on changes on numerical attributes in the presence of key dependencies and wrt denial constraints, while keeping the numerical distance to the original database to a minimum. This framework introduces new challenging decision and optimization problems, and many algorithmic and complexity theoretic issues. We concentrate in particular on the "Database Repair Problem" (*DRP*) of determining the existence of a repair at a distance not greater than a given bound. In particular, we consider the problems of construction and verification of such a repair. These problems are highly relevant for large inconsistent databases. For example, solving *DRP* can help us find the minimum distance from a repair to the original instance. This information can be used to prune impossible branches in the process of materialization of a repair. The *CQA* problem of deciding the consistency of query answers is studied wrt decidability, complexity, and approximation under several alternative semantics.

We prove that *DRP* is *NP*-complete for denial constraints, which are enough to capture census-like applications. *CQA* belongs to Π_2^P and becomes Δ_2^P-hard. For a particular, simple, but relevant class of denials we get tractability of CQA for a large and relevant class of non-aggregate queries. For the same class of denials, simple aggregations based on acyclic conjunctive queries easily lead to intractability of CQA.

Wrt approximation, we prove that *DRP* is *MAXSNP*-hard in general; and for a relevant subclass of denials, we provide a polynomial time approximation within a constant factor. All the algorithmic and complexity results, unless otherwise stated, refer to data complexity [8], i.e. to the size of the database that here includes a binary representation for numbers. For complexity theoretic definitions and classical results we refer to [9].

Moving to the case of real numbers would certainly bring new issues that would require different approaches. They are left for ongoing and future research. Actually, it would be natural to investigate them in the richer context of constraint databases [10].

For databases like those we are considering here, aggregation constraints may also be relevant. Here we briefly study the *DRP* and *CQA* problems when repairs have to satisfy aggregation constraints. It turns out that both problems become undecidable when both the instance and the constraints are part of the input.

This paper is structured as follows. Section 2 introduces basic definitions. Section 3 presents the notion of database repair, several semantics for the notion of consistent answer to a query; and some relevant decision problems. Section 4 investigates their complexity. In Section 5, approximation algorithms for the problem of finding the minimum distance to a repair are studied. We obtain negative results for the general case, but a good approximation for the relevant class of *local* denial constraints. Section 6 investigates tractability of *CQA* for conjunctive queries and denial constraints containing one database atom plus built-ins. Section 7 contains extensions of the main framework, like a brief analysis of repairs that have to satisfy certain statistical conditions, the above mentioned results around aggregation constrains, and a discussion of alternative distances. Section 8 presents some conclusions and refers to related work.

2. Preliminaries

Consider a relational schema $\Sigma = (\mathscr{U}, \mathscr{R}, \mathscr{B})$, with domain $\mathscr{U}$ that includes $\mathbb{Z}$, $\mathscr{R}$ a set of database predicates, $\mathscr{B}$ a set of built-in predicates. If a predicate $R \in \mathscr{R}$ has arity $n \geqslant 1$, each of its n arguments has associated a unique attribute name that is not shared with other argument positions of database predicates in the schema. This is not an essential restriction, but it will make the formulation of some definitions much simpler. The set of attribute names (we will simply call them attributes) in the schema is denoted with $\mathscr{A}$. According to this, we usually denote a database predicate with $R(A_1, \ldots, A_n)$, with each $A_i \in \mathscr{A}$. $\mathscr{A}(R)$ denotes the set of attributes of R. It holds $\mathscr{A}(R) \subseteq \mathscr{A}$.

Each attribute A has a domain that is a subset of $\mathscr{U}$, where it can take values. Different attributes may share the same domain and make take the same values. Numerical attributes are those that have domain $\mathbb{Z}$. With denial constraints, we can make a numerical attribute take values in a subset of $\mathbb{Z}$, e.g. in $\mathbb{N}$ or $\{0, 1\}$. For the latter case, we can use denial constraints like $\forall x, y \neg(R(x, y) \wedge x < 0)$, $\forall x, y \neg(R(x, y) \wedge x > 1)$.

A database instance is a finite collection D of *database tuples*, i.e. of ground atoms $R(\bar{c})$, with $R \in \mathscr{R}$ and $\bar{c}$ a finite sequence of constants in $\mathscr{U}$. If $R(A_1, \ldots, A_n) \in \mathscr{R}$, $t = R(c_1, \ldots, c_n) \in D$, and $S = (A_{i_1}, \ldots, A_{i_k})$ is a subsequence of $(A_1, \ldots, A_n)$, then $t[A_{i_1}, \ldots, A_{i_k}]$ denotes the projection of tuple t on S, i.e. $(c_{i_1}, \ldots, c_{i_k})$. For $k = 1$, we simple write $t(A_i) = c_i$.

There is a set $\mathscr{F} \subseteq \mathscr{A}$ containing all the *fixable* attributes, those that take values in $\mathbb{Z}$ and are allowed to be fixed. Attributes outside $\mathscr{F}$ are called *rigid*. $\mathscr{F}$ need not contain all the numerical attributes, that is, we may also have rigid numerical attributes. More precisely, each predicate $R \in \mathscr{R}$ has a set of fixable attributes, denoted by $\mathscr{F}(R)$. It holds $\mathscr{F}(R) \subseteq \mathscr{F}$. We also have a set $\mathscr{K}$ of key constraints expressing that predicates $R \in \mathscr{R}$ have a primary key K_R, with $K_R \subseteq (\mathscr{A}(R) \backslash \mathscr{F}(R))$. Later on (cf. Definition 2), we will assume that $\mathscr{K}$ is satisfied both by the initial instance D, denoted $D \vDash \mathscr{K}$, and its repairs. In this sense, we say the elements of $\mathscr{K}$ are *hard*. Since $\mathscr{F}(R) \cap K_R = \emptyset$, values in rigid attributes cannot be changed in a repair process. In addition, there may be a separate set IC of *flexible* ICs that may be violated, and it is the job of a repair to restore consistency wrt them (while still satisfying $\mathscr{K}$).

A *linear denial constraint* [10] has the form $\forall \bar{x} \neg(A_1 \wedge \cdots \wedge A_m)$, where the A_i are database atoms (i.e. with predicate in $\mathscr{R}$), or built-in atoms of the form $x\theta c$, where x is a variable, c is a constant, and $\theta \in \{=, \neq, <, >, \leqslant, \geqslant\}$, or of the form $x = y$. If $x \neq y$ is allowed, we call them *extended* linear denials. We assume that all the constants appearing in ICs belong to the domain $\mathscr{U}$. In a constraint, $\bar{x}$ denotes the sequence of variables, say $\bar{x} = x_1, \ldots, x_n$, that appear in the conjunction of atoms. Since the order in which the variables appear in the quantification does not matter, we usually identify $\bar{x}$ with the set formed by its variables, say $\{x_1, \ldots, x_n\}$. Furthermore, in denials we usually replace $\wedge$ by a comma, and sometimes we use $\overline{\forall}$ for the whole prefix of universal quantifications. *Unless otherwise stated, all the flexible ICs in this paper are denial constraints; and sets of denials are always finite.*

Example 2. The following are linear denials: (a) No customer is younger than 21: $\forall Id, Age, Income, Status \neg(Customer(Id, Age, Income, Status), Age < 21)$. (b) No customer with income less than 60 000 has "silver" status: $\forall Id, Age, Income, Status \neg(Customer(Id, Age, Income, Status), Income < 60\,000, Status = silver)$. (c) The constraint in Example 1, i.e. $\forall T, L, Type, Flow \neg(Traffic(T, L, Type, Flow), Type = 0, Flow > 1000)$.

In this example, in order to make the intuitive contents of a denial constraint more clear, we have used the attribute names as variables. Sometimes this practice will allow us to simplify the formulation of some definitions and results. This can always be done, by introducing extra versions of the attributes names if necessary; versions that are not shared by other attribute names. For example, for the predicate $R(A, B)$, the denial $\forall x, y, z \neg(R(x, z), R(y, z), z = 1)$ can be rewritten as $\forall A, A', B \neg(R(A, B), R(A', B), B = 1)$.

We will consider aggregate queries containing the aggregation functions *sum*, *count*, or *average*. More precisely, an *aggregate conjunctive query* has the form $q(x_1, \ldots, x_m; agg(z)) \leftarrow B(x_1, \ldots, x_m, z, y_1,$

$\ldots, y_n)$, where *agg* is an aggregation function. The *non-aggregate matrix* (NAM) of the aggregate query is given by $q'(x_1,\ldots,x_m) \leftarrow B(x_1,\ldots,x_m,z,y_1,\ldots,y_n)$, which is a usual first-order (FO) conjunctive query with built-in atoms. In the query predicate q, the *aggregation attribute* (or variable) z does not appear among the x_i. We use the set semantics for aggregate queries. An aggregate conjunctive query is *cyclic* (*acyclic*) if its NAM is cyclic (acyclic) [8].

Example3. $q(x,y,sum(z)) \leftarrow R(x,y), Q(y,z,w), w \neq 3$ is an aggregate conjunctive query, with aggregation attribute z. Each answer (x,y) to its NAM, i.e. to $q(x,y) \leftarrow R(x,y), Q(y,z,w), w \neq 3$, is expanded to $(x,y,sum(z))$ as an answer to the aggregate query. $sum(z)$ is the sum of all the values for z having a w, such that (x,y,z,w) makes $R(x,y), Q(y,z,w), w \neq 3$ true. In the database instance $D = \{R(1,2), R(2,3), Q(2,5,9), Q(2,6,7), Q(3,1,1), Q(3,1,5), Q(3,8,3)\}$ the answer set for the aggregate query is $\{(1,2,5+6),(2,3,1+1)\}$. In this example, the aggregate query is a *group-by* query, because the query predicate has free variables (x and y).

An *aggregate comparison query* is a sentence of the form $q(agg(z)) \wedge agg(z)\theta k$, where $q(agg(z))$ is the head of a *scalar* aggregate conjunctive query (i.e. with no free variables, or equivalently, without group-by), θ is a comparison operator, and k is an integer number. For example, the following is an aggregate comparison query asking whether the aggregated value obtained via $q(sum(z))$ is greater than 5: $Q : q(sum(z)) \wedge sum(z) > 5$, with $q(sum(z)) \leftarrow R(x,y), Q(y,z,w), w \neq 3$. We can see that aggregate comparison queries are boolean, i.e. they have a *true* or *false* answer in a database instance. An aggregate comparison query $q(agg(z)) \wedge agg(z)\theta k$ is (a)cyclic if the NAM of the query that defines $q(agg(z))$ is (a)cyclic.

3. Least squares repairs

When we update numerical values to restore consistency, it is desirable to make the smallest overall variation of the original values, while considering the relative relevance or specific scale of each of the fixable attributes. Since the original instance and a repair will share the same rigid values (cf. Definition 2), we can use them to compute variations in the numerical values. Now, we make this idea more precise.

We say that instances D, D' over Σ are *rigid-comparable* if for every tuple $t = R(\bar{c}) \in D$, for some $R \in \mathscr{R}$, there is a unique tuple $t' = R(\bar{c}') \in D'$ such that $t[\mathscr{A}(R)\backslash\mathscr{F}(R)] = t'[\mathscr{A}(R)\backslash\mathscr{F}(R)]$, and vice versa. In this case, we write $t' = m(t)$, indicating that tuple $t' \in D'$ is the corresponding version of $t \in D$, possibly modified at its fixable attributes. That is, tuples t and t' coincide on the values of their rigid attributes.

Definition 1. For rigid-comparable instances D and D' over schema Σ, their *square distance* is $\Delta_{\bar{\alpha}}(D,D') = \sum_{t \in D, A \in \mathscr{F}} \alpha_A \cdot (t(A) - m(t)(A))^2$, and $\bar{\alpha} = (\alpha_A)_{A \in \mathscr{F}}$.

Definition 2. Let D, D' be instances over the same schema Σ, such that $D \vDash \mathscr{K}$ and $D' \vDash \mathscr{K}$; and IC be a set of flexible ICs. D' is a *repair* for D wrt IC if: (a) D, D' are rigid-comparable; and (b) $D' \vDash IC$. A *least squares repair* (LS-repair) for D is a repair D' that minimizes the square distance $\Delta_{\bar{\alpha}}(D,D')$ over all the instances that satisfy (a) and (b).

The conditions in this definition make D and D' rigid-comparable, and Definition 1 can be applied. In general, we are interested in LS-repairs, but (not necessarily minimum) repairs will be useful auxiliary instances.

Example 4 (*Example 1 continued*). $\mathscr{R} = \{Traffic\}$, $\mathscr{A} = \{Time, Link, Type, Flow\}$, $K_{Traffic} = \{Time, Link\}$, $\mathscr{F} = \{Type, Flow\}$, with weights $\bar{\alpha} = (10^{-5}, 1)$, resp. A repair of the original instance D is $D_1 = \{Traffic(1.1,a,0,1000), Traffic(1.1,b,1,900), Traffic(1.3,b,1,850)\}$. In this case, $m(Traffic(1.1,a,0,1100)) = Traffic(1.1,a,0,1000)$, etc. Another repair is $D_2 = \{Traffic(1.1,a,1,1100), Traffic(1.1,b,1,900), Traffic(1.3,b,1,850)\}$. The distances are $\Delta_{\bar{\alpha}}(D,D_1) = 100^2 \times 10^{-5} = 10^{-1}$ and $\Delta_{\bar{\alpha}}(D,D_2) = 1^2 \times 1$. D_1 is the only LS-repair.

The coefficients α_A can be chosen in many different ways depending on factors like relative relevance of attributes, actual distribution of data, measurement scales, etc. In the rest of this paper we will assume, for simplification, that $\alpha_A = 1$ for all $A \in \mathscr{F}$, and $\Delta_{\bar{\alpha}}(D,D')$ will be simply denoted by $\Delta(D,D')$.

Example 5. Database D has predicates $Client(ID,A,C)$, with attributes for identification (the key), age and credit line of the client; and $Buy(ID1,I,P)$, with key $\{ID1,I\}$ and containing clients buying items at certain prices. There are two denial ICs $ic_1 : \forall ID1, P, A, C \neg(Buy(ID1,I,P), Client(ID,A,C), ID1 = ID, A < 18, P > 25)$ and $ic_2 : \forall ID, A, C\ \neg(Client(ID,A,C), A < 18, C > 50)$, requiring that people younger than 18 cannot spend more than 25 on one item nor have a credit line higher than 50 in the store. The following table shows the database contents. We added an extra column to be able to refer to the tuples.

D:

Client	*ID*	*A*	*C*		*Buy*	*ID1*	*I*	*P*	
	1	15	52	t_1		1	CD	27	t_4
	2	16	51	t_2		1	DVD	26	t_5
	3	60	900	t_3		3	DVD	40	t_6

We can see that ic_1 is violated by $\{t_1, t_4\}$ and $\{t_1, t_5\}$, and ic_2 by $\{t_1\}$ and $\{t_2\}$. Assuming that $\alpha_A = \alpha_C = \alpha_P = 1$, we have two LS-repairs, D', D'', at a distance 10 from the original instance.

D':

Client	*ID*	*A*	*C*		*Buy*	*ID1*	*I*	*P*	
	1	15	**50**	t_1'		1	CD	**25**	t_4'
	2	16	**50**	t_2'		1	DVD	**25**	t_5'
	3	60	900	t_3		3	DVD	40	t_6

D'':

Client	*ID*	*A*	*C*		*Buy*	*ID1*	*I*	*P*	
	1	**18**	52	t_1''		1	CD	27	t_4
	2	16	**50**	t_2''		1	DVD	26	t_5
	3	60	900	t_3		3	DVD	40	t_6

In this example, it was possible to obtain LS-fixes by performing direct, local changes in the original conflictive tuples alone. No new, intermediate inconsistencies are introduced in the repair process. The following example shows that this may not be always the case.

Example 6. Consider a database D with relations $P(A, B, C)$ and $Q(D, E)$ with $K_P = \{A\}$, $K_Q = \{D\}$ and $\mathscr{F} = \{B, C, E\}$; and linear denials $ic_1 : \forall x, y, z, w \neg(P(x, y, z), Q(x, w), y > 3, w > 5)$ and $ic_2 : \forall x, y, z \neg (P(x, y, z), y < 5, z < 4)$. The following instance is inconsistent, because $\{t_1, t_2\}$ violates ic_1.

P	*A*	*B*	*C*		*Q*	*D*	*E*	
	a	6	1	t_1		a	9	t_2

If we tried to find a repair by making the smallest change that restores consistency, $t_1(B)$ would be replaced by 3 (the alternative of replacing $t_2(E)$ by 5 is more expensive):

P	*A*	*B*	*C*		*Q*	*D*	*E*	
	a	**3**	1	t_1'		a	9	t_2

This new instance is still inconsistent since $\{t_1'\}$ violates ic_2. Now, if we tried to solve this new inconsistency by making the smallest variation, $t_1'(B)$ would be replaced by 5, which violates ic_2 again. Actually, the only LS-repair is

P	*A*	*B*	*C*		*Q*	*D*	*E*	
	a	6	1	t_1		a	**5**	t_2'

We can see that new inconsistencies can be introduced by local changes, and that an LS-repair is not necessarily a sequence of minimum local changes.

The numerical values in the denial constraints define threshold values that may determine their satisfaction by a database instance.

Definition 3. Let IC be a set of extended linear denials. The set of *borders* of IC is $Borders(IC) = \{c \mid$ there is $ic \in IC$ and a variable x in ic, such that x appears in the position of fixable attribute in a predicate in ic and either $(x < c)$, $(x > c)$, $(x \leqslant c-1)$, $(x \geqslant c+1)$, $(x = c-1)$, $(x = c+1)$, or $(x \neq c)$ appears as a comparison in $ic\}$.

When comparisons of the type $x = y$ or $x \neq y$, and joins involve only rigid attributes, the built-in atoms in extended linear denials determine a solution space for repairs as an intersection of semi-spaces. LS-repairs can be found taking values from the borders of the ICs (cf. previous examples). However, if comparisons of the type $x = y$ or $x \neq y$, or joins in the denials involve fixable attributes, then the attribute values in LS-repairs can be found in intervals around borders, as defined by the denials, and around values in tuples of the inconsistent database.

Lemma 1. *Let D' be an LS-repair of D wrt a set IC of extended linear denial constraints. Let $a = |\mathscr{A}|$. For each tuple $t \in D'$ and fixable attribute A of t, it holds $t(A) \in [v - a, v + a]$, for some integer v in a tuple in D, or $t(A) \in [c - a, c + a]$, for some $c \in Borders(IC)$. Furthermore, if IC is such that equality and $\neq$ atoms between attributes, and joins* *involve only rigid attributes, then either* $t(A) = m^{-1}(t)(A)$ *or* $t(A) \in Borders(IC)$.[1]

If there are equalities, non-equalities or joins involving fixable attributes, the LS-repairs can take values that are not borders nor values from the inconsistent database. The following examples show such cases and illustrate Lemma 1.

Example 7. Consider a predicate $T(X, Y)$, with only Y fixable, and the instance $D = \{T(3,3)\}$, which is inconsistent wrt $IC = \{\forall x, y \neg(T(x,y), x = y)\}$. In this case, $Borders(IC) = \emptyset$. There are two LS-repairs: $D' = \{T(3,2)\}$ and $D'' = \{T(3,4)\}$. The values for attribute Y in both LS-repairs can be found in the interval $[v - |\mathscr{A}|, v + |\mathscr{A}|]$, with $v = 3$ and $|\mathscr{A}| = 2$.

Example 8. Consider a predicate $R(A, B, C, D)$, with only B fixable, and a set of ICs $IC = \{\forall x, y, z, w \neg (R(x,y,z,w), y<6), \forall x, y, z, w \neg(R(x,y,z,w), y = z), \forall x, y, z, w\ \neg(R(x,y,z,w), z = w)\}$. The instance $D = \{t\}$ with $t : R(1,5,6,7)$ is inconsistent. Here, $Borders(IC) = \{6\}$. The inconsistency wrt the first IC can be solved by changing $t(B)$ from 5 to 6. This new value violates the second constraint. So now, to stay as close as possible to the original instance, we replace $t(B)$ by 7. This value now violates the third constraint. By increasing the value by one once more, we get an LS-repair $D' = \{R(1,8,6,7)\}$. The value taken by B is a border value plus 2.

Proof of Lemma 1. First we will concentrate in the case where IC is such that the attributes participating in equality atoms between attributes or in joins are all rigid. Let us assume, by contradiction, that there exists a tuple $t \in D'$ and a fixable attribute A, for which $t(A) = k$ and $k \neq m(t)(A)$ and $k \notin Borders(IC)$. Without loss of generality, assume that $k < m(t)(A)$. Let D'' be the same as D' except that $t(A)$ is changed from k to $k-1$. Since $k \notin Borders(IC)$, no new inconsistencies can be added since the built-ins that were (or were not) satisfied before, will continue in the same state. Thus, D'' also satisfies IC and is rigid-comparable to D. However, D'' is closer to D than D'. Thus, D' is not an LS-repair. We have a contradiction.

In the general case, ICs may have fixable attributes participating in (non-)equality atoms between attributes or in joins. In this case, the value of such an attribute in a tuple might be changed to satisfy a constraint, so that the equality does not hold. As before, we assume there exists an LS-repair D' and an attribute A such that $t(A) = k$ and k is not in $[v - a, v + a]$ for each numerical value v in a tuple in D, nor in $[c - a, c + a]$ for each $c \in Borders(IC)$. In a similar way as in the proof for the case with equalities and joins between rigid attributes, a contradiction can be reached. □

It is easy to construct examples with an exponential number of repairs. For the kind of repairs and ICs we are considering, it is possible that no repair exists, in contrast to [2,11], where, if the set of ICs is consistent as a set of logical sentences, a repair for a database always exists.

Example 9. $R(A, B)$ has key A, and B is fixable. $IC = \{\forall x_1, x_2, y \neg(R(x_1,y), R(x_2,y), x_1 = 1, x_2 = 2), \forall x_1, x_2, y \neg(R(x_1,y), R(x_2,y), x_1 = 1, x_2 = 3), \forall x_1, x_2, y \neg(R(x_1,y), R(x_2,y), x_1 = 2, x_2 = 3), \forall x, y \neg(R(x,y), y > 3), \forall x, y \neg(R(x,y), y < 2)\}$ is consistent. The first three ICs force attribute B to be different in every tuple. The last two ICs require $2 \leqslant B \leqslant 3$. The inconsistent database $D = \{R(1,-1), R(2,1), R(3,5)\}$ has no repair.

Proposition 1. *If D has a repair wrt IC, then it also has an LS-repair wrt IC.*

Proof. Let ρ be the square distance between D and a repair D' according to Definition 1. The circle of radius ρ around D containing instances over the same schema that share the rigid attribute values with D intersects the non empty "consistent" region that contains the database instances with the same schema and rigid values as D and satisfy IC. All the instances within that circle have their fixable numerical values bounded in absolute value by a fixed function of ρ and the fixable values in D. In consequence, the circle has a finite number of instances, and the distance takes a minimum in the consistent region. □

4. Decidability and complexity

In applications where repairs are based on changes of numerical values, computing concrete repairs is a relevant problem. In databases containing census forms, correcting the latter before doing statistical processing is a common problem [3]. In databases with experimental samples, we can fix certain erroneous quantities as specified by linear ICs. In these cases, the repairs are relevant objects to compute explicitly, which contrasts with CQA [2],

[1]We recall that $m(t) \in D'$, with D' a repair of D, is the tuple that results from modifying $t \in D$.

where the main motivation for introducing repairs is to formally characterize the notion of a consistent answer to a query, as an answer that persists under all possible repairs. In consequence, we now consider some decision problems related to existence and verification of LS-repairs, and to CQA under different semantics.

Definition 4. For an instance D and a set IC of ICs:

(a) $Rep(D, IC) = \{D' | D'$ is an LS-repair of D wrt $IC\}$, the *repair checking problem*.

(b) $Rep(IC) = \{(D, D') | D' \in Rep(D, IC)\}$.

(c) $NE(IC) = \{D | Rep(D, IC) \neq \emptyset\}$, for *non-empty* set of repairs, i.e. the problem of *checking existence of LS-repairs*.

(d) $DRP(IC) = \{(D, k) |$ there is $D' \in Rep(D, IC)$ with $\Delta(D, D') \leqslant k\}$, the *DRP*, i.e. the problem of checking existence of LS-repairs within a given positive distance k.

(e) $DROP(IC)$ is the optimization problem of finding the minimum distance from an LS-repair wrt IC to a given input instance.

Notice that, by Proposition 1, $DRP(IC)$ could also be defined as $\{(D, k) |$ there is a repair D' of D with $\Delta(D, D') \leqslant k\}$.

Definition 5. Let D be a database, IC a set of ICs, and Q a conjunctive query.[2] (a) A finite sequence $\bar{c}$ of constants in $\mathcal{U}$ is a *consistent answer* to $Q(\bar{x})$ under the: (a1) *skeptical semantics* if for every $D' \in Rep(D, IC)$, $D' \models Q(\bar{c})$; (a2) *brave semantics* if there exists $D' \in Rep(D, IC)$ with $D' \models Q(\bar{c})$;[3] (a3) *majority semantics* if $|\{D' | D' \in Rep(D, IC)$ and $D' \models Q(\bar{c})\}| > |\{D' | D' \in Rep(D, IC)$ and $D' \not\models Q(\bar{c})\}|$.

(b) That $\bar{c}$ is a consistent answer to Q in D under semantics $\mathfrak{S}$ is denoted by $D \models_{\mathfrak{S}} Q[\bar{c}]$. If Q is boolean (i.e. a sentence) and $D \models_{\mathfrak{S}} Q$, we say that *yes* is a consistent answer, meaning that Q is true in the repairs of D as prescribed by semantics $\mathfrak{S}$. $CA(Q, D, IC, \mathfrak{S})$ is the set of consistent answers to Q in D wrt IC under semantics $\mathfrak{S}$. For a boolean Q, if $CA(Q, D, IC, \mathfrak{S}) \neq \{yes\}$, $CA(Q, D, IC, \mathfrak{S}) = \{no\}$.

(c) $CQA(Q, IC, \mathfrak{S}) = \{(D, \bar{c}) | \bar{c} \in CA(Q, D, IC, \mathfrak{S})\}$ is the decision *problem of CQA*, i.e. of checking consistent answers.

[2]Whenever we say "conjunctive query", we mean a non-aggregate query.

[3]Skeptical and brave semantics are aka. *certain* and *possible* semantics, resp.

In the literature on CQA, the notion of consistent answer and the CQA problem usually refer to the skeptical semantics.

Proposition 2. *$NE(IC)$ can be reduced in polynomial time to the complements of $CQA(False, IC, Skeptical)$ and CQA $(True, IC, Majority)$, where False, True are ground queries that are always false, resp. true.*

Proof. First for the skeptical semantics. Given a database instance D, consider the instance (D, no) for $CQA(False, IC, Skeptical)$, corresponding to the question "Is there an LS-repair of D wrt IC that does not satisfy *False*?" has answer *yes* iff the class of LS-repairs of D is empty. For the majority semantics, for the instance (D, no) for $CQA(True, IC, Majority)$, corresponding to the question "Is it not the case that the majority of the LS-repairs satisfy *True*?", we get answer *yes* iff the set of LS-repairs is empty. □

In Proposition 2, it suffices for queries *True*, *False* to be true, resp. false, in all instances on the same schema as the input database. The former can be represented by $(yes) \leftarrow$, a query with empty body; and the latter by $(yes) \leftarrow R(\ldots, x, \ldots), x = 1, x = 2$, where variable x corresponds to a numerical attribute.

Theorem 1. *For every fixed set IC of linear denials: (a) Deciding if for an instance D there is a repair D' with $\Delta(D, D') \leqslant k$, with positive integer k that is part of the input, is in NP. (b) There is a fixed set IC of denials for which DRP(IC) is NP-complete.*

Proof. (a) 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 that appear in built-in predicates in the explicit versions of the ICs. For example, consider the denial $\forall x, y, z \neg(P(x, y), Q(z, x), y > 2)$. The explicit version is $\forall x, y, z, w \neg(P(x, y), Q(z, w), y > 2, x = w)$ and l would be 3 (for x, y, w).

Notice that, by Proposition 1, there is a repair at a distance not greater than k iff there is an LS-repair at a distance not greater than k.

If there exists an LS-repair D' with $\Delta(D, D') \leqslant k$, then no value in a fixable attribute for a tuple in D'

differs from its corresponding value in D by more than $\sqrt{k}$. In consequence, the size of an LS-repair may not differ from the original instance by more than $l \times n \times bin(k)/2$, where $bin(k)$ is the size of the binary representation of k. Thus, the size of an LS-repair 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.

(b) Membership: According to Proposition 1, there is an LS-repair at a distance $\leqslant k$ iff there is a repair D' at a distance $\leqslant k$. We use part (a) of this proposition.

Hardness: We can reduce *Vertex Cover* to $DRP(IC_0)$ for a fixed set of denials IC_0. Given a graph instance $(\mathscr{V}, \mathscr{E})$, k for VC, consider a database schema with a binary predicate $E(X, Y)$ with key $\{X, Y\}$ for the edges of the graph, and a predicate $V(X1, Chosen)$, with key $X1$ that takes vertices as values. Attribute *Chosen* is the only fixable attribute. The original database D contains the tuples $E(e_1, e_2), E(e_2, e_1)$ for $\{e_1, e_2\} \in \mathscr{E}$, and the tuples $V(v, 0)$ for $v \in \mathscr{V}$. The constraint IC: $\forall x, y, c_1, c_2 \neg(E(x,y) \wedge V(x, c_1) \wedge V(y, c_2) \wedge c_1 < 1 \wedge c_2 < 1)$ expresses that for any edge, at least one of the incident vertices is covered. A vertex cover of size k exists iff there exists an LS-repair of D wrt IC at a distance $\leqslant k$. The encoding is polynomial in the size of the original graph. □

By Proposition 1, there is a repair for D wrt IC at a distance $\leqslant k$ iff there is an LS-repair at a distance $\leqslant k$. So, a test for the former, that is analyzed in Theorem 1(a), can be used for the latter. Actually, if we happen to know, e.g. by using the test in Theorem 1(a), that there is a repair at a distance $\leqslant k$, then the minimum distance between D and a repair (i.e. the distance between D and any LS-repair) can be found by binary search in the distance interval $[0, k]$ in $\log(k)$ steps, using at each of them the test in Theorem 1(a).

If an LS-repair exists, its square distance to D is polynomially bounded by the size of D (cf. Lemma 2 below). Since D and a repair have the same number of tuples, only the size of their values in a repair matter, and they are constrained by a fixed set of linear denials and the condition of minimality.

Lemma 2. *Given a database D and a set of extended denials IC, the size of an LS-repair D′ is polynomial in the size of D and the numerical constants in the ICs. This is also true if D′ is a repair obtained from D by replacing values of fixable attributes by values at the intervals around numerical values in D or the borders determined by the ICs (according to Lemma 1).*

Proof. The proof of the first claim follows immediately from: (a) The LS-repair D' has the same number of tuples as D, and (b) by Lemma 1, in an LS-repair, the value for each attribute and in each tuple falls in an interval of the form $[c - |\mathscr{A}|, c + |\mathscr{A}|]$, where c is either a value of D or a border. For the second claim, the proof is similar, since D' has the same number of tuples as D. □

Theorem 2. *For a fixed set IC of extended linear denials*: (a) *The problem NE(IC) of deciding if an instance has an LS-repair wrt IC is NP-complete, and* (b) *CQA under the skeptical and the majority semantics is coNP-hard.*

Proof. (a) For hardness, it suffices to consider linear denials. We reduce 3-*Colorability* to $NE(IC_0)$, for a fixed set IC_0 of ICs. Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be an undirected graph with set of vertices $\mathscr{V}$ and set of edges $\mathscr{E}$. Consider the following database schema, instance D, and set IC_0 of ICs:

1. Predicate $Vertex(Id, Red, Green, Blue)$, with key Id, and domain $\mathbb{N}$ for the last three attributes, actually the only three fixable attributes in the schema. For each $v \in \mathscr{V}$, we have $Vertex(v, 0, 0, 0)$ in D (and no other *Vertex* tuple in D).
2. Predicate $Edge(Id1, Id2)$, with no fixable attributes. For each $e = \{v_1, v_2\} \in \mathscr{E}$, $Edge(v_1, v_2)$, $Edge(v_2, v_1) \in D$.
3. Predicate $Tester(Red1, Green1, Blue1)$, with no fixable attributes, and extension $Tester(1,0,0), Tester(0,1,0), Tester(0,0,1)$ in D.
4. ICs:

$$\forall i, x, y, z \neg(Vertex(i, x, y, z), x < 1, y < 1, z < 1);$$

$$\forall i, x, y, z \neg(Vertex(i, x, y, z), x > 1) \text{ (the same for } y, z);$$

$$\forall i, x, y, z \neg(Vertex(i, x, y, z), x = 1, y = 1, z = 1);$$

$$\forall i, x, y, z \neg(Vertex(i, x, y, z), x = 1, y = 1); \text{etc.}$$

$$\forall i, j, x, y, z \neg(Vertex(i, x, y, z), Vertex(j, x, y, z), Edge(i, j), Tester(x, y, z)).$$

If there is an LS-repair wrt IC_0 of the generated instance, then the graph is 3-colorable. If the graph is 3-colorable, then there is a consistent instance with the same rigid values as the original instance.

Thus, by Proposition 1, there is an LS-repair. The reduction is polynomial in the size of the graph.

Now we prove membership. By Proposition 1 and Lemma 1, the existence of a repair is equivalent to the existence of an LS-repair; and the latter is equivalent to the existence of a repair that has its modified fixable values taken at the intervals around the borders of the corresponding attributes or around the values in the tuples of D (cf. Lemma 1). So, we can concentrate on the latter problem. An NP algorithm for this problem is as follows: (1) For the positive cases D, guess a witness, i.e. a repair D' of D that shares the values of non-fixable attributes with D, and the modified values of the fixable attributes taken from intervals of the form $[c-|\mathscr{A}|, c+|\mathscr{A}|]$, for c a value in D or a border. (2) Check that D, D' are rigid-comparable. (3) Check that D' satisfies the ICs. This test is polynomial in the size of D, D'. By Lemma 2, the size of D' is polynomial in the size of D.

(b) *coNP*-hardness follows from Proposition 2 and part (a). $\square$

For hardness in (a) and (b) in Theorem 2, linear denials suffice. Membership in (a) can be obtained for any fixed finite set of extended denials.

Theorem 3. *For a fixed set IC of linear denials*: (a) *The problem Rep(IC) of checking if an instance is an LS-repair is coNP-complete*, and (b) *CQA under skeptical semantics is in Π_2^P, and, for ground atomic queries, Δ_2^P-hard.*

Proof. (a) We reduce 3-*SAT*'s complement to *Rep*(*IC*) for a fixed schema and set *IC* of denials. We have a predicate $Lit(l, \bar{l})$ whose extension stores complementary literals (only), e.g. $Lit(p, \neg p)$ when p is one of the variables in the instance Φ of SAT. Also a predicate Cl for tuples of the form $Cl(\varphi, l, k)$, where φ is a clause of Φ (we assume, wlog. that all clauses have exactly three literals), l is a literal in the clause, and k takes value 0 or 1 (for the truth value of l in φ). The first two arguments are the key of Cl. Finally, we have a predicate $Aux(K, N)$, with key K and fixable numerical attribute N, and a predicate $Num(N1)$ with a rigid numerical attribute $N1$.

Consider an instance $\Phi = \varphi_1 \wedge \cdots \wedge \varphi_m$ for 3-SAT. We produce an instance D for the predicates as indicated above, assigning arbitrary truth values to the literals in Cl, but making sure that, in the whole set of Cl-tuples, a literal takes only one truth value, and complementary literals take complementary truth values. We also have $Aux(0,0)$, and $Num(\lceil\sqrt{s+1}\rceil)$ in D, where s is the number of different pairs of the form (φ_i, l), with l a literal that appears in φ_i. There are no other *Aux*- or *Num*-tuples in D. $\square$

Consider now the following set of denials:

(a) $\overline{\forall}\neg(Cl(\varphi,l,u), u>1)$; $\overline{\forall}\neg\ (Cl(\varphi,l,u), u<0)$.
(b) $\overline{\forall}\neg(Cl(\varphi,l,u), Cl(\psi,l,v), u\neq v)$.
(c) $\overline{\forall}\neg(Cl(\varphi,l,u), Cl(\psi,l',v), Lit(l,l'), u=v)$.
(d) $\overline{\forall}\neg(Cl(\varphi,l,u), Cl(\varphi,l',v), Cl(\varphi,l'',w), Aux(k,n), l\neq l', l\neq l'', l'\neq l'', u=v=w=0, n=0)$.
(e) $\overline{\forall}\neg(Num(z), Aux(k,n), n\neq 0, n\neq z)$.

Denial (a) indicates that 0, 1 are possible truth values. Denial (b), that a literal takes only one truth value in the whole set of Cl-tuples. Denial (c) indicates that complementary literals take different truth values. Denial (d), that each clause becomes true, and then also Φ, or *Aux* takes a value other than 0 in its second attribute. Finally, denial (e) indicates that the value in the second attribute of *Aux* has to be 0 or $\lceil\sqrt{s+1}\rceil$. Attribute K in *Aux* is introduced just to have a key. Other than this, it is not relevant.

If Φ is unsatisfiable, then the original instance is inconsistent, because (d) is violated. Even more, in this case there is no repair that can be obtained by changing truth values only, because (d) would still be violated. In this case, in order to make (d) true, only the value of the second attribute of *Aux* has to be changed, from 0 to $\lceil\sqrt{s+1}\rceil$, as prescribed by (e). The distance between this repair and D is $(\lceil\sqrt{s+1}\rceil)^2$, which is greater than s. This is the closest repair to D we can have when Φ is not satisfiable.

When Φ is satisfiable, it may be the case that D already encodes a satisfying truth assignment. In this case, D is consistent and it is only LS-repair, at a distance 0. If D does not encode a satisfying assignment, but there is one, we can change the truth values in the Cl-tuples in D in order to encode in a repair D' of D the satisfying assignment. In this case, the distance between D and D' is at most s (this generous upper bound is reached only if all literals have to change their truth values).

It holds that Φ is unsatisfiable iff the instance D' that coincides with D except for *Aux*, that now contains only $Ans(0, \lceil\sqrt{s+1}\rceil)$ instead, is an LS-repair of D wrt *IC*. Thus, checking D' for LS-repair of D suffices to check unsatisfiability.

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

(b) For the first claim on CQA, let IC and a query Q be given. The complement of CQA is in NP^{coNP}: Given an instance D, nondeterministically choose an instance D' with $D' \nvDash Q$ and a repair D' of D. The latter test can be done in *coNP* (by part (a)). But $NP^{coNP} = NP^{\Sigma_1^P} = \Sigma_2^P$. In consequence, CQA belongs to $co\Sigma_2^P = \Pi_2^P$.

For the second claim, we prove hardness of *CQA* by a *LOGSPACE*-reduction from the following problem [12, Theorem 3.4]: Given a Boolean formula in 3CNF $\Phi(p_1, \ldots, p_n)$, decide if the last variable p_n is equal to 1 in the lexicographically maximum satisfying assignment (the answer is *No* if Φ is not satisfiable).

We consider a fixed database schema containing predicate $Var(V, T, Weight)$, with key V and fixable attributes T, taking values 0 or 1, and $Weight$. It also contain predicate $Cl(C, Var_1, Val_1, Var_2, Val_2, Var_3, Val_3)$, with key C and no fixable attributes.

Now, if Φ is $\varphi_1 \wedge \cdots \wedge \varphi_m$, with each φ_i a clause, we create an instance D as follows. For each variable p_i, $Var(p_i, 0, 2^{n-i})$ goes into D. In binary encoding, the values 2^{n-i} are polynomial in the size of original formula. For each clause $\varphi_i = l_{i_1} \vee l_{i_2} \vee l_{i_3}$, $Cl(\varphi_i, p_{i_1}, \tilde{l}_{i_1}, p_{i_2}, \tilde{l}_{i_2}, p_{i_3}, \tilde{l}_{i_3})$ is inserted into D, where $\tilde{l}_{i_j}$ is equal to 1 in case of positive occurrence of variable p_{i_j} in φ_i and equal to 0 for a negative occurrence. For example, for $\varphi_6 = p_6 \vee \neg p_9 \vee p_{12}$, $Cl(\varphi_6, p_6, 1, p_9, 0, p_{12}, 1)$ is inserted.

We consider the following ICs[4]:

(a) $\forall v, t \neg(Var(v, t, \_) \wedge t < 0)$;
$\forall v, t \neg(Var(v, t, \_) \wedge t > 1)$.
(b) $\forall v, t, w \neg(Var(v, t, w) \wedge t = 0 \wedge w > 0)$.
(c) $\forall c, v_1, x_1, v_2, x_2, v_3, x_3, u_1, u_2, u_3 \neg(Cl(c, v_1, x_1, v_2, x_2, v_3, x_3) \wedge Var(v_1, u_1, \_) \wedge Var(v_2, u_2, \_) \wedge Var(v_3, u_3, \_) \wedge x_1 \neq u_1 \wedge x_2 \neq u_2 \wedge x_3 \neq u_3)$.

The idea is that the 3rd, 5th and 7th arguments in *Cl*-tuples contain the truth value that makes the propositional variable in the preceding argument true. According to denial (c), the truth values in the 2nd argument of *Var*-tuples associated to a same clause cannot differ all from the right value prescribed by the corresponding *Cl*-tuple. In this way we make the clause true. The extended denial constraint in (c) could be replaced by eight non-extended denial constraints.

Instance D is inconsistent due to (b). Each repair of D represents a satisfying assignment for Φ. If Φ is not satisfiable, there is no repair of D. If it is, in order to obtain a satisfying assignment, the values in the 2nd argument of *Var* have to be changed to obtain a repair.

Let us now consider the square distance from a repair to D. Each repair D' is associated to a satisfying truth assignment $S = \langle s_1, \ldots, s_m \rangle$ for $\langle p_1, \ldots, p_n \rangle$. If $i_1 < \cdots < i_r$ is the sequence of all the indices in S associated to 0's in S, the square distance from D' to D is $2^{2(n-i_1)} + 2^{2(n-i_2)} + \cdots + 2^{2(n-i_r)} + (n - r)$, because due to (b) we have to give value 0 to *Weight* for each variable that retains the value 0 it had in D. The term $(n - r)$ comes from the truth values that were changed from 0 to 1.

Assume that $S = \langle s_1, \ldots, s_n \rangle$ and $S' = \langle t_1, \ldots, t_n \rangle$ are satisfying truth assignments (for $\langle p_1, \ldots, p_n \rangle$) with $S \prec S'$ under the lexicographical order. In this case, there exists an integer m such that $0 = s_m < t_m = 1$, while for all $k < m$, $s_k = t_k = 0$. We can compare the square distances to D from the repairs $D(S), D(S')$, associated to S, S', resp. Since for m it holds $s_m = 0$ and $t_m = 1$, the tuple $Var(p_m, 0, 2^{(n-m)})$ in D has to be changed to $Var(p_m, 0, 0)$ in $D(S)$, contributing to the square distance with $2^{2(n-m)} + 1$, which is greater than the sum of terms for higher indices (and smaller exponents) with which S' may contribute to the distance $\Delta(D, D(S'))$. Notice that for both $D(S)$ and $D(S')$, the sums of the first $m - 1$ terms of the distance (corresponding to the first $(m - 1)$ indices) are the same, namely $\Sigma_{i=1}^{m-1} 2^{2(n-i)}$.

We can see that $S \prec S'$ implies $\Delta(D, D(S')) < \Delta(D, D(S))$. In consequence, the closest repair to D in square distance (i.e. the only LS-repair if any) corresponds to the maximum satisfying assignment for Φ in the lexicographical order. It is good enough to check if this repair has p_n taking value 1: The consistent answer to the ground atomic query $Var(p_n, 1, 1)$ is *yes* iff p_n takes the value 1 in the lexicographically maximum satisfying truth assignment.

Membership in Theorem 3(a) can be obtained for any set of extended denials. It is still open to close the gap between the lower and upper data complexity bounds for CQA.

[4]In underscore, _, in an argument of an atom means that any fresh variable may appear at its place.

Theorem 4. *For aggregate comparison queries using sum, CQA under linear denials and brave semantics is coNP-hard.*

Proof. A reduction from *Vertex Cover* 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 predicates $Ver(V, Z), Edge(V1, V2)$, where V is a key for Ver, and Z is the only fixable attribute, that takes values in $\{0, 1\}$, which can be enforced by including in IC_0 the linear denials $\forall x, z \neg(Ver(x, z), z > 1)$, $\forall x, z \neg(Ver(x, z), z < 0)$. Intuitively, Z indicates with 1 if the vertex V is in the cover, and with 0 otherwise. The values for the attributes of *Edge* are vertices and then, non-numerical.

In the original database D we have the tuples $Ver(e, 0)$, for $e \in \mathcal{V}$; and also the tuples $Edge(e_1, e_2), Edge(e_2, e_1)$ for $\{e_1, e_2\} \in \mathcal{E}$. Putting into IC_0 the linear constraint $\forall x_1, z_1, x_2, z_2 \neg (Ver(x_1, z_1), Ver(x_2, z_2), \quad Edge(x_1, x_2), z_1 < 1, z_2 < 1)$, the LS-repairs of the database are in one-to-one correspondence with the vertex covers of minimum cardinality.

For the query $Q^{(k)} : q(sum(z)) \wedge sum(z) < k$, with $q(sum(z)) \leftarrow Ver(x, z)$, the instance (D, yes) for CQA under brave semantics has answer *No*, (i.e. $Q^{(k)}$ is false in all LS-repairs) only for every k smaller than the minimum cardinality c of a vertex cover. □

5. Approximation for the DRP

We consider the problem of finding a good approximation for the optimization problem *DROP*(*IC*).

Proposition 3. *For a fixed set IC of linear denials, DROP(IC) is MAXSNP-hard.*

Proof. By reduction from the *MAXSNP*-hard problem *B-Minimum Vertex Cover* (BMVC) which asks for a minimum vertex cover in a graph whose nodes have a bounded degree [13, Chapter 10]. We encode the graph as in the proof of Theorem 4. We also use the same initial database D. Every LS-repair 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 *DRP*(*IC*) [9]. □

As an immediate consequence [9], we obtain that *DROP*(*IC*) cannot be uniformly approximated within an arbitrarily small constant factor.

Corollary 1. *There is a set IC of linear denials for which, unless P = NP, there is no Polynomial Time Approximation Schema for DROP(IC).*

This negative result does not preclude the possibility of finding an efficient algorithm for approximation within a constant factor for *DROP*. Actually, in the following we do this for a restricted but still useful and interesting class of denial constraints.

5.1. Local denials

Definition 6. A set of linear denials *IC* is *local* if[5]: (a) Attributes participating in equality atoms between attributes or in joins are all rigid. (b) There is a built-in atom with a fixable attribute in each element of *IC*. (c) No element of *IC* contains $\neq$ involving a fixable attribute. (d) No attribute A appears in *IC* both in comparisons of the form $A < c_1$ and $A > c_2$.

Without loss of generality and for simplicity of the presentation, we assume in what follows that each local constraint contains (a) $<$ and $>$, but not $\leqslant$ or $\geqslant$[6]; (b) at most one comparison of the type $A\theta c$ per attribute A, for θ either $>$ or $<$.

Example 10. The denial $\forall x, y, z \neg(R(x, y, z), y > 3, y > 5, z \leqslant 7)$ has the second attribute compared more than once, and the third attribute is compared with a $\leqslant$. The denial can be replaced by $\forall x, y, z \neg(R(x, y, z), y > 5, z < 8)$.

In Example 5, *IC* is local. In Example 6, the set of ICs is not local since attribute B of relation P is compared through both $<$ and $>$. In Example 9, *IC* is not local for the same reason. Local constraints have the property that by solving a particular inconsistency, no new inconsistencies are generated as shown in the following example.

Example 11 (*Example 5 continued*). The ICs are local. IC ic_1 is violated by $\{t_1, t_4\}$ and $\{t_1, t_5\}$, and ic_2 by $\{t_1\}$ and $\{t_2\}$. The first inconsistency can be solved by updating $t_4(P)$ from 27 to 25. The resulting instance is such that ic_1 is violated by $\{t_1, t_5\}$, and ic_2 by $\{t_1\}$ and $\{t_2\}$. No new inconsistency is introduced.

Example 12. Consider a predicate $R(A, B, C)$, where A is the key and C is the fixable attribute; and the local ICs $ic_1 : \forall x, y, z \neg(R(x, y, z), z > 4)$, and

[5]We assume here that attributes or versions thereof are used as variables in the ICs.

[6]The comparisons $x \leqslant c$ and $x \geqslant c$ can be expressed as $x < c + 1$ and $x > c - 1$, resp.

$ic_2: \forall x, y, z \neg(R(x,y,z), z > 2)$. The instance D that contains only the tuple $t: R(a,1,5)$ violates both ic_1 and ic_2. The inconsistency wrt ic_1 can be solved by replacing $t(C)$ by 4. In the new database, the updated tuple t' is still inconsistent wrt ic_2, but no new inconsistencies arise from the update.

Lemma 3. *Given an instance D and a set IC of local denial constraints, there always exists an LS-repair of D wrt IC.*

Proof. Since IC is local, in each denial in it, there is at least one fixable attribute involved in a built-in. Changing its value, the comparison atom can be falsified and the whole constraint can be satisfied. We have to show that such a change can be made for the whole set of ICs. For each fixable attribute A in IC, we are able to derive an interval $[c_A, \infty)$ or $(-\infty, c_A]$ such that if we pick up the value of A from it, all the ICs involving A in a built-in are satisfied. An interval of the form $[c_A, \infty)$ can be found if A is compared with $<$, and of the form $(-\infty, c_A]$ if it is compared with $>$. Let $\mathscr{F}_{\mathscr{B}}$ be the set of fixable attributes that are in a built-in atom in at least one element of IC.

An instance D' can be constructed from D by replacing the value of every fixable attribute $A \in \mathscr{F}_{\mathscr{B}}$ in every tuple $t \in D$ by c_A. This new instance satisfies the constraints, therefore it is a repair of D wrt IC. By Proposition 1, there is an LS-repair. □

Locality is a sufficient, but not necessary condition for existence of LS-repairs. This can be seen with the database $D = \{P(a,2)\}$, whose first attribute is the key, and the non-local set of denials $\{\forall x, y \neg(P(x,y), y < 3), \forall x, y \neg(P(x,y), y > 5)\}$. D has $\{P(a,3)\}$ as LS-repair. *In the rest of Section* 5 *we will assume that the sets of flexible denials associated to a schema are local.*

Proposition 4. *There is a set IC of local denials, such that DRP(IC) is NP-complete, and DROP(IC) is MAXSNP-hard.*

Proof. For the first claim, membership follows from Theorem 1(b). For hardness, we can do the same reduction as in Theorem 1(b), because the ICs used there are local denials. For the second claim, we can proceed as in the proof of Proposition 3, but instead of using the non-local denials in that proof, we can use the single local denial $ic: \forall x_1, z_1, x_2, z_2 \neg (Ver(x_1, z_1), Ver(x_2, z_2), \quad Edge(x_1, x_2), z_1 < 1, z_2 < 1)$. We need no denials that restrict the attributes to take values in $\{0, 1\}$, because in the tuples in D, attribute Z takes value 0. By minimality of LS-repairs, when restoring consistency wrt ic, Z will only be modified to take value 1. □

This proposition tells us that the problem of finding good approximations in the case of local denials is still relevant. Local constraints do not make our decision problems easier. For example, Theorem 4 still holds for them, because in its proof the first denial in IC_0 can be eliminated, and the two remaining form a local set. This is due to the contents of the initial instance and the minimality imposed on value changes.

Definition 7. Let $ic \in IC$ be denial constraint of the form

$$\forall \bar{x} \neg(A_1(\bar{x}_1) \wedge \cdots \wedge A_m(\bar{x}_m) \wedge B_1(\bar{x}_{m+1}) \wedge \cdots \wedge B_{m+s}(\bar{x}_{m+s})), \tag{1}$$

where $\bar{x} = \bigcup_{j=1}^{m} \bar{x}_j \supseteq \bigcup_{j=m+1}^{m+s} \bar{x}_j$, the A_j are database predicates and the B_j are built-in predicates.

(a) A set $I = \{A'_1(\bar{a}_1), \ldots, A'_k(\bar{a}_k)\}$ of (ground) database atoms is a *violation set* for ic in instance D iff: (a1) $A'_j(\bar{a}_j) \in D$, $j = 1, \ldots, k$. (a2) There is a substitution $\theta: \bar{x} \rightarrow \mathscr{U}$, such that $\{A_1(\bar{x}_1), \ldots, A_m(\bar{x}_m)\}\theta = I$, and, for $i = m+1, \ldots, m+s$, $B_i(\bar{x}_i)\theta$ is true in $\mathscr{U}$.

(b) The set of labeled violation sets of IC in D is $\mathscr{I}(D, IC) = \{(I, ic) | ic \in IC \text{ and } I \text{ is a violation set for } ic\}$.

Here, $A(\bar{x}')\theta$, with $\bar{x}' \subseteq \bar{x}$, is the ground atom that results from applying θ to the variables in $A(\bar{x}')$; and $\{A_1(\bar{x}_1), \ldots, A_m(\bar{x}_m)\}\theta = \{A_1(\bar{x}_1)\theta, \ldots, A_m(\bar{x}_m)\theta\}$. In consequence, the substitution θ in the definition is a unifier of $\{A_1(\bar{x}_1), \ldots, A_m(\bar{x}_m)\}$ and I. So, $k \leqslant m$, and the predicates A'_j must be among the A_i in (1). Notice that a violation set makes the conjunction in (1) true in D, and then ic becomes false in D. $\mathscr{I}(D, IC)$ contains the violation sets together with the constraintthey violate, which is used as a label. In this way, the same set of tuples can be associated to different constraints. Elements (I, ic) of $\mathscr{I}(D, IC)$ will still be called violations *sets*. Notice that the definition of violation set can be applied to any extended denial.

Example 13 (*Example 5 continued*). A violation set for ic_1 is $\{t_1, t_4\}$ since, for $\theta(ID) = 1$, $\theta(A) = 15$, $\theta(C) = 25$, $\theta(I) = CD$ and $\theta(P) = 27$, it holds $\{Buy(ID1, I, P), Client(ID, A, C)\}\theta = \{t_1, t_4\}$, and both $(A < 18)\theta$ and $(P > 25)\theta$ are true. Similarly, $\{t_1, t_5\}$ is also a violation set for ic_1, and $\{t_1\}$ and $\{t_2\}$ are both violation sets for ic_2. In consequence, $\mathcal{I}(D,IC) = \{(\{t_1,t_4\},ic_1),(\{t_1,t_5\},ic_1),(\{t_1\},ic_2),(\{t_2\},ic_2)\}$.

Notice that the *conflict hypergraph* introduced in [14] for studying classic CQA wrt denial constraints has as vertices the database tuples in D; and as hyperedges, the violation sets for elements ic of IC. In our case, each hyperedge is labelled with its corresponding ic. If the denial constraints are functional dependencies, we obtain *conflict graphs* [11].

Example 14 (*Example 13 continued*). The conflict hypergraph as shown in Fig. 1 contains four hyperedges, those corresponding to the violation sets $(\{t_1,t_4\},ic_1)$, $(\{t_1,t_5\},ic_1)$, $(\{t_1\},ic_2)$ and $(\{t_2\},ic_2)$.

Definition 8. (a) Consider an instance D and a set IC of ICs. Let t be a tuple in D such that $t \in I \subseteq D$, where (I,ic) is a violation set for $ic \in IC$ in D. A database tuple t' is a *local repair* of t (wrt I and ic) if: (a1) t' uses the same database predicate as t. (a2) t' has the same values as t in all the attributes but one fixable attribute. (a3) Replacing t by t' solves the inconsistency wrt ic, i.e. $((I\backslash\{t\}) \cup \{t'\},ic)$ is not a violation set of IC in $(D\backslash\{t\}) \cup \{t'\}$. (a4) There is no tuple t'' that simultaneously satisfies (a1)–(a3), differs from t on the same attribute as t', and $\Delta(\{t\},\{t''\}) < \Delta(\{t\},\{t'\})$, where Δ denotes quadratic distance.

(b) $S(t,t') = \{(I,ic) | ic \in IC, t \in I$ and $(I\backslash\{t\}) \cup \{t'\}$ is not a violation set for ic in $(D\backslash\{t\}) \cup \{t'\}\}$.

A local repair t' of t solves the violation of at least one IC where t participates, minimizes the distance from t, and differs from t in the value of only *one* attribute. Thus, if a tuple t is consistent, i.e. it does not belong to any violation set, then it has no local repairs. It holds $S(t,t') \subseteq \mathcal{I}(D,IC)$, and the former set contains the violation sets that include t and are solved by replacing t' by t. We will usually apply the notation $S(t,t')$ to a tuple t and one of its local repairs t'.

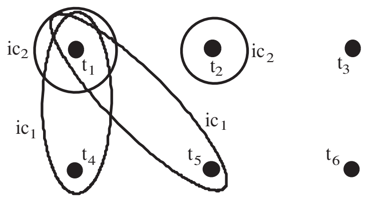

Fig. 1. Conflict hypergraph.

The attribute that has been changed by a local repair t' of t is denoted by $adiff(t,t')$.[7] In the examples, we will usually write in bold the attribute values that are modified by local repairs.

Example 15 (*Example 13 continued*). Tuple t_1' : $Client(1,15,\mathbf{50})$ is a local repair of t_1, because: (a) it modifies only the value of the fixable attribute C in t_1; (b) for ic_2, replacing t_1 by t_1' solves the violation set $(\{t_1\},ic_2)$; and (c) there is no other tuple that solves the same violation set and is closer to t_1. In this case, $adiff(t_1,t_1') = C$, and $S(t_1,t_1') = \{(\{t_1\},ic_2)\}$.

Tuple t_1'' : $Client(1,\mathbf{18},52)$ is also a local repair of t_1, with $S(t_1,t_1'') = \{(\{t_1,t_4\},ic_1),(\{t_1,t_5\},ic_1),(\{t_1\},ic_2)\}$. Tuples t_2, t_4 and t_5 have one local repair each, namely t_2' : $Client(2,16,\mathbf{50})$, t_4' : $Buy(1,CD,\mathbf{25})$, and t_5' : $Buy(1,DVD,\mathbf{25})$, respectively, with $S(t_2,t_2') = \{(\{t_2\},ic_2)\})$, $S(t_4,t_4') = \{(\{t_1,t_4\},ic_1)\})$ and $S(t_5,t_5') = \{(\{t_1,t_5\},ic_1)\})$, respectively. The consistent tuple t_3 has or needs no local repair.

5.2. DRP as a set cover problem

For a fixed sets IC of local denials, we can solve the instances of $DROP(IC)$ by transforming them into instances of a *minimum weighted set cover optimization problem* ($MWSCP$). This problem is $MAXSNP$-hard [9,15], and its general approximation algorithms approximate within a logarithmic factor [15,16]. By concentrating on local denials, we will be able to generate versions of the $MWSCP$ that can be approximated within a constant factor (cf. Section 5.3).

Definition 9. For a database D and a set IC of local denials, the instance $(U,\mathcal{S},w)$ for the $MWSCP$, where U is the underlying set, $\mathcal{S}$ is the collection of subsets of U, and w is the weight function,[8] is given by: (a) $U = \mathcal{I}(D,IC)$; (b) $\mathcal{S} = \{S(t,t') | t'$ is a local repair for a tuple $t \in D\}$; and (c) $w(S(t,t')) = \Delta(\{t\},\{t'\})$.

By construction, all the $S(t,t')$ in the MWSCP are non-empty. Also, since the ICs are local, the union of $\mathcal{S}$ is U. This holds since for each $(I,ic) \in U$, ic contains at least one fixable attribute A in a built-in,

[7]We recall that an attribute is associated to a unique database predicate and only one of its arguments.

[8]In the corresponding $MWSCP$, we try to find (the weight of) a $\mathcal{C} \subseteq \mathcal{S}$, such that, for every $u \in U$, there is $s \in \mathcal{C}$ with $u \in s$ (i.e. a set cover for U), and $\mathcal{C}$ has minimum weight (given by the sum of the weights of its elements).

and there is tuple $t \in I$ such that, by modifying $t(A)$ we can get a local repair t' of t such that $I \in S(t, t')$.

Example 16 (*Examples 5 and 15 continued*). We illustrate this reduction from *DROP* to *MWSCP*, and give an idea about how we are going to use minimum set covers to obtain LS-repairs. Here, $U = \{(\{t_1, t_4\}, ic_1), (\{t_1, t_5\}, ic_1), (\{t_1\}, ic_2), (\{t_2\}, ic_2)\}$, and $\mathscr{S} = \{S(t_1, t_1'), S(t_1, t_1''), S(t_2, t_2'), S(t_4, t_4'), S(t_5, t_5')\}$. The contents of each of the elements in $\mathscr{S}$ is shown in the table below. Elements of $\mathscr{S}$ are the columns, and their elements, the rows. An entry 1 means that the set of $\mathscr{S}$ contains the corresponding element in the first column; and a 0, otherwise.

Set	$S(t_1, t_1')$	$S(t_1, t_1'')$	$S(t_2, t_2')$	$S(t_4, t_4')$	$S(t_5, t_5')$
Weight	4	9	1	4	1
$(\{t_1, t_4\}, ic_1)$	0	1	0	1	0
$(\{t_1\}, ic_2)$	1	1	0	0	0
$(\{t_1, t_5\}, ic_1)$	0	1	0	0	1
$(\{t_2\}, ic_2)$	0	0	1	0	0

A minimum set cover for this problem is $\mathscr{C}_1 = \{S(t_1, t_1''), S(t_2, t_2')\}$, with total weight 10. It is a minimum cover, because the union of its elements is U, and there is no other cover with less weight. $\mathscr{C}_1$ shows that, by replacing t_1 by t_1'' and t_2 by t_2', all the violation sets are solved, i.e. they are no violation sets for the same constraint anymore. Actually, the database D' obtained from D by replacing t_1 by t_1'' and t_2 by t_2' is an LS-repair of D. These replacements lead to a consistent database for two reasons: (1) Since $S(t_1, t_1'')$ and $S(t_2, t_2')$ together form a cover, all the inconsistencies are solved in D'; and (2) Since the constraints are local, no new inconsistencies are generated by these replacements. If the constraints were not local, the replacements would solve the initial inconsistencies, but might introduce new ones. Cf. Definition 11 and Lemma 5 below for a formal treatment of this idea.

Another minimum cover, therefore also with weight 10, is $\mathscr{C}_2 = \{S(t_1, t_1'), S(t_2, t_2'), S(t_4, t_4'), S(t_5, t_5')\}$. The database D'' obtained from D by replacing t_1 by t_1', t_2 by t_2', t_4 by t_4', and t_5 by t_5' is also an LS-repair.

As we can see, we could think of constructing an LS-repair by replacing each inconsistent tuple $t \in D$ by a local repair t' with $S(t, t') \in \mathscr{C}$, where $\mathscr{C}$ is a minimum set cover for the corresponding instance $(U, \mathscr{S}, w)$ of the *MWSCP*. The problem is that for a tuple t, it might be the case that $S(t, t')$ and $S(t, t'')$ belong both to the minimum set cover $\mathscr{C}$. In that case, it is not clear how to construct the LS-repair by replacing t.

Example 17. Consider a schema with a predicate $R(A, B, C, D)$, with $K_R = \{A\}$, $\mathscr{F}(R) = \{B, C\}$. The set IC of local denials contains $ic_1 : \bar{\forall}\neg(R(x, y, z, w), y > 3)$, $ic_2 : \bar{\forall}\neg(R(x, y, z, w), y > 5, w > 7)$, and $ic_3 : \bar{\forall}\neg(R(x, y, z, w), z < 4)$. The database instance D containing only the tuple $t : R(1, 6, 1, 8)$ is inconsistent. Here, $\mathscr{I}(D, IC) = \{(\{t\}, ic_1), (\{t\}, ic_2), (\{t\}, ic_3)\}$. There are three local fixes, namely, $t_1 : R(1, \mathbf{3}, 1, 8)$, $t_2 : R(1, \mathbf{5}, 1, 8)$ and $t_3 : R(1, 6, \mathbf{4}, 8)$. For them, $S(t, t_1) = \{(\{t\}, ic_1), (\{t\}, ic_2)\}$, $S(t, t_2) = \{(\{t\}, ic_2)\}$ and $S(t, t_3) = \{(\{t\}, ic_3)\}$. The instance of *MWSCP* is

Set	$S(t, t_1)$	$S(t, t_2)$	$S(t, t_3)$
Weight	9	1	9
$(\{t\}, ic_1)$	1	0	0
$(\{t\}, ic_2)$	1	1	0
$(\{t\}, ic_3)$	0	0	1

The only minimum cover is $\mathscr{C} = \{S(t, t_1), S(t, t_3)\}$. In this case, we could attempt to obtain an LS-repair by replacing t by both t_1 and t_3, but this would result in a violation of the key constraint on R. However, the local repairs t_1 and t_3 can be combined into a new tuple $t_4 = R(1, \mathbf{3}, \mathbf{4}, 8)$, which is not a local repair, but solves all the inconsistencies. The database D' obtained by replacing t by t_4 is an LS-repair.

Our next results tells us that the replacement we made in the previous example is always possible.

Lemma 4. *Let $\mathscr{C}$ be a minimum cover for instance $(U, \mathscr{S}, w)$ of the MWSCP associated to D and IC. For different local repairs t', t'' of t such that $S(t, t'), S(t, t'') \in \mathscr{C}$, it holds $adiff(t, t') \neq adiff(t, t'')$.*

Proof. Let us assume, by contradiction, that there are $S(t, t'), S(t, t'') \in \mathscr{C}$ such that $adiff(t, t') = adiff(t, t'') = A$ and $t'(A) < t''(A)$. Since the ICs are local, either A is compared in all ICs in IC with either $<$ or $>$. Without loss of generality, we assume the latter. Since $t'(A) < t''(A)$, it is easy to see that $S(t, t'') \subseteq S(t, t')$. Thus, $\mathscr{C} \backslash \{S(t, t'')\}$ is also a cover. This implies that $\mathscr{C}$ is not minimum, and we have a contradiction. $\square$

This result may not hold if the cover is not minimum. This lemma allows us to combine in one tuple the local repairs that participate in a minimum cover. These combined tuple repairs will be used to construct a new consistent database.

Definition 10. Let $\mathscr{C}$ be a minimum cover for instance $(U, \mathscr{S}, w)$ of the *MWSCP* associated to D, IC. (a) Let $t_1, \ldots, t_n$ be the local repairs of $t \in D$, such that $S(t, t_i) \in \mathscr{C}$, for $i \in [1, n]$. The *combined local repair* of t, denoted $t^\star$, is such that $t^\star(A) = t_i(A)$ when $A = \mathit{adiff}(t, t_i)$, and $t^\star(A) = t(A)$, otherwise.

(b) If $T^\star = \{(t, t^\star) |$ there is t' such that $S(t, t') \in \mathscr{C}\}$, then we define $D(\mathscr{C}) = \bigcup_{(t,t^\star) \in T^\star} \ [(D \backslash \{t\}) \cup \{t^\star\}]$.

$D(\mathscr{C})$ is the database instance obtained from D by replacing t by $t^\star$ whenever $(t, t^\star) \in T^\star$. Notice that $t^\star$ may not be a local repair of t, because it may change more that one attribute. However, $t^\star$ is obtained from a set of tuples that modify only one attribute of t each. Whenever we have a cover $\mathscr{C} \subseteq \mathscr{S}$, where all the local repairs t' in the elements $S(t, t')$ of $\mathscr{C}$ change different attributes values of t, we may compute the $t^\star$'s, no matter if $\mathscr{C}$ is minimum or not. After that, $T^\star$ and its corresponding new instance $D(\mathscr{C})$ can be computed as in Definition 10(b).

Example 18 (*Example 17 continued*). The combined local repair $t^\star$ for t is obtained from the local repairs represented in $\mathscr{C} = \{S(t, t_1), S(t, t_3)\}$: $t^\star$ becomes $R(1, \mathbf{3}, \mathbf{4}, 8)$. Thus, $T^\star = \{(t, R(1, \mathbf{3}, \mathbf{4}, 8))\}$, and $D(\mathscr{C}) = \{R(1, \mathbf{3}, \mathbf{4}, 8)\}$, which is in fact the only LS-repair of D.

Now we can establish that there is a one-to-one correspondence between the minimum covers of the *MWSCP* and the LS-repairs.

Theorem 5. *If $\mathscr{C}$ is a minimum cover for instance $(U, \mathscr{S}, w)$ of the MWSCP associated to D, IC, then $D(\mathscr{C})$ is an LS-repair of D wrt IC, and $\Delta(D, D(\mathscr{C})) = w(\mathscr{C})$. Furthermore, for every LS-repair D' of D wrt IC, there exists a minimum cover $\mathscr{C}$ for the instance $(U, \mathscr{S}, w)$ of the MWSCP associated to D and IC, such that $D' = D(\mathscr{C})$.*

In order to prove this theorem we need first some auxiliary concepts and technical lemmas. First, we will prove that local repairs do not introduce new inconsistencies when the denials are local.

Definition 11. Consider an instance D and a set of local denials IC:

(a) For (I, ic) a violation set for D and IC, define $I[\mathscr{A} \backslash \mathscr{F}] = \{(R, t[\mathscr{A}(R) \backslash \mathscr{F}(R)]) |$ there are $R \in \mathscr{R}$ and $\bar{c}$ with $t = R(\bar{c}) \in I\}$.
(b) $\mathscr{I}(D, IC)[\mathscr{A} \backslash \mathscr{F}] = \{(I[\mathscr{A} \backslash \mathscr{F}], ic) | (I, ic) \in \mathscr{I}(D, IC)\}$.
(c) Let t, t' be database tuples such that $\{t\}$ and $\{t'\}$ are rigid-comparable as instances, and $t \in D$. Replacing t by t' in D *does not generate new inconsistencies* if $\mathscr{I}(D', IC)[\mathscr{A} \backslash \mathscr{F}] \ \backslash \ \mathscr{I}(D, IC)[\mathscr{A} \backslash \mathscr{F}] = \emptyset$, where $D' = (D \backslash \{t\}) \cup \{t'\}$.

$I[\mathscr{A} \backslash \mathscr{F}]$ denotes the set of tuples of constants obtained from database tuples t in I by projecting the t's on their rigid attributes, and then annotating them with their predicate names R (to keep track of their origin).

Example 19 (*Example 12 continued*). In this case, $IC = \{ic_1, ic_2\}$, t is $R(a, 1, 5)$, t' is $R(a, 1, 4)$, $\mathscr{I}(D, IC) = \{(\{R(a, 1, 7)\}, ic_1), (\{R(a, 1, 7)\}, ic_2)\}$. For $D' = \{R(a, 1, 4)\}$, $\mathscr{I}(D', IC) = \{(\{R(a, 1, 4)\}, ic_2)\}$.

To check if new inconsistencies are generated, we compute the difference between $\mathscr{I}(D', IC)[\mathscr{A} \backslash \mathscr{F}] = \{(\{(R, (a, 1))\}, ic_2)\}$ and $\mathscr{I}(D, IC)[\mathscr{A} \backslash \mathscr{F}] = \{(\{(R, (a, 1))\}, ic_1), (\{(R, (a, 1))\}, ic_2)\}$. Since the difference is empty, replacing t by its local repair t' does not generate new inconsistencies.

Lemma 5. *If t' is a local repair of a tuple t wrt instance D and a set of local denials IC, then replacing t by t' in D does not generate new inconsistencies.*

Proof. Let $\mathit{adiff}(t, t') = A$. Since IC is local, attribute A can only appear in IC in $<$- or $>$-atoms, but not both. Without loss of generality, assume the latter is the case. So, $t'(A) < t(A)$.

For $D' = (D \backslash \{t\}) \cup \{t'\}$, we need to prove that $(\mathscr{I}(D', IC)[\mathscr{A} \backslash \mathscr{F}] \backslash \mathscr{I}(D, IC)[\mathscr{A} \backslash \mathscr{F}]) = \emptyset$. By contradiction, assume that for $ic \in IC$, there is a violation set $(I', ic) \in \mathscr{I}(D', IC)$ such that there is no $(I, ic) \in \mathscr{I}(D, IC)$ with $I[\mathscr{A} \backslash \mathscr{F}] = I'[\mathscr{A} \backslash \mathscr{F}]$. Since the rigid values are kept in a repair, this is equivalent to saying that there is a violation set $(I', ic) \in \mathscr{I}(D', IC)$ for which $(m^{-1}(I'), ic) \notin \mathscr{I}(D', IC)$. Since t' is the only difference between D and D', it holds $t' \in I'$. But $t'(A) < t(A)$, the only difference between D and D' is attribute A, and in all the constraints attribute A is compared only with $>$. Therefore, if $(I', ic) \in \mathscr{I}(D', IC)$, then $(m(I'), ic) \in \mathscr{I}(D, IC)$. We have a contradiction. $\square$

The following lemma justifies that an LS-repair can always be constructed from a set of local repairs that are combined as described in Definition 10. This lemma will allow us to prove later that, for every LS-repair D', there is minimum cover $\mathscr{C}$ such that $D' = D(\mathscr{C})$.

Lemma 6. *Consider an LS-repair D' of D wrt IC, and $L(D,D') = \{(t,t') | t,t'$ are R-tuples for some $R \in \mathscr{R}, t \in D$, and there are an R-tuple $t'' \in (D' \backslash D)$ and $A \in \mathscr{A}(R)$ such that $t = m^{-1}(t''), t''(A) \neq t(A), t'(A) = t''(A)$; and, for each $B \neq A, t'(B) = t(B)\}$. It holds*: (a) *For each $(t,t') \in L(D,D')$, t' is a local repair of t.* (b) *$\mathscr{C}(D,D') = \{S(t,t') | (t,t') \in L(D,D')\}$ is a cover of the MWSCP associated to D and IC.* (c) *$D(\mathscr{C}(D,D')) = D'$.*

For each tuple $t \in D$, the tuple $m(t)$ in D' (the t'' in the definition of $L(D,D')$) may differ in more than one attribute value from t. The set $L(D,D')$ contains the pairs (t,t'), such that t' coincides with the modified version of t, i.e. $m(t)$, in repair D' in exactly one modified attribute value, but coincides with t at the other attributes. That is, we are decomposing the modified versions of tuples in D into its "local repair components". Thus, $L(D,D')$ traces back and finds the set of local repairs that can transform D into D'.

There will be as many local repairs of t in $L(D,D')$ as attribute values in t have been changed by $m(t)$. If the local repairs obtained from $L(D,D')$ are combined as described in Definition 10, we would obtain a set $T^\star$ that is needed to transform D into D'. For different local repairs t' and t'' such that $(t,t'),(t,t'') \in L(D,D')$, it holds $adiff(t,t') \neq adiff(t,t'')$. Thus, it is possible to compute $D(\mathscr{C}(D,D'))$, even without proving that the cover $\mathscr{C}(D,D')$ is minimum. However, later on we will show it actually is (cf. proof of Theorem 5).

Example 20 (*Example 18 continued*). The only LS-repair of D, that contains only tuple $t : R(1,6,1,8)$, wrt IC is $D' = \{R(1,\mathbf{3},\mathbf{4},8)\}$. This repair is obtained from D by two local repairs: one changing attribute B from 6 to 3 and another changing attribute C from 1 to 4. Thus, $L(D,D') = \{(t,t'),(t,t'')\}$, with $t' : R(1,\mathbf{3},1,8)$ and $t'' : R(1,6,\mathbf{4},8)$.

The repair D' can be reconstructed from $L(D,D')$ by combining the local repairs as described in Definition 10. In fact, $\mathscr{C}(D,D') = \{S(t,t'), S(t,t'')\}$, and $T^\star = \{(t,t^\star)\}$, with $t^\star = R(1,\mathbf{3},\mathbf{4},8)$. Therefore, $D(\mathscr{C}(D,D')) = \{R(1,\mathbf{3},\mathbf{4},8)\} = D'$.

Proof of Lemma 6. (a) Let $(t,t') \in L(D,D')$. By the minimality of D' as an LS-repair, the change of value in attribute $A = adiff(t,t')$ has to solve some inconsistency. More precisely, there are $ic \in IC$ and $I \subseteq D$, such that (I,ic) is a violation set wrt D and IC, $t \in I$, and t' solves the violation set, i.e. $((I \backslash \{t\}) \cup \{t'\}, ic)$ is not a violation set wrt $(D \backslash \{t\}) \cup \{t'\}$. We collect in a set all these violation sets. So, consider $S = S(t,t')$, the set of all the violation sets that are solved by replacing t by t'. S is non-empty, because $(I,ic) \in S$. Furthermore, t' clearly satisfies conditions (a1)–(a3) for being a local repair of t through every (I,ic) in S. We have to check that (a4) holds for at least one element of S.

Consider $IC' = \{ic \in IC |$ there is I with $(I,ic) \in S\}$. Let us assume that attribute A appears in IC in comparisons of the type $>$ (the case of comparisons with $<$ is similar). Let c be the smallest constant with which A is compared through $A > c$ in IC'. Let (I_0, ic_0) be an element of S such that $A > c$ appears in ic_0. Notice that for t to belong to the violation sets in S, it must hold $t(A) > c$.

We have three cases: $t'(A) > c$, $t'(A) < c$, and $t'(A) = c$. If $t'(A) > c$, then (I_0, ic_0) cannot be solved by replacing t by t'. Since $(I_0, ic_0) \in S$, we have a contradiction. If $t'(A) < c$, we can construct the instance $D'' = (D' \backslash \{t'\}) \cup \{m(t)'\}$, where $m(t)'$ coincides with $m(t)$ (the modified version of t in D') except at attribute A, for which $m(t)'(A) = c$. The same inconsistencies will be solved by D'' since c is the smallest value that appears in a comparison involving A. It holds $\Delta(\{D\},\{D''\}) < \Delta(\{D\},\{D'\})$; which contradicts the LS-minimality of D'. Thus, it must be $t'(A) = c$. In this case condition (a4) holds for (I_0, ic_0), because any tuple t'' with $t''(A) > c$, i.e. closer to the original instance, will not solve the violation set (I_0, ic_0). In consequence, t' is a local repair of t for (I_0, ic_0).

(b) Now, we will prove that $\mathscr{C}(D,D') = \{S(t,t') | (t,t') \in L(D,D')\}$ is a cover of the MWSCP. We have already proven that every $S(t,t') \in \mathscr{C}(D,D')$ is such that t' is a local repair of t. Since D' is an LS-repair of D wrt IC, the set of local repairs needed to construct it solve all the inconsistencies, and therefore $\mathscr{C}(D,D')$ is a cover. □

Proof of Theorem 5. We start proving the first statement, i.e. that the instance associated to a minimum cover is an LS-repair. Since the union of all the elements of a cover $\mathscr{C}$ is U, the local repairs represented by the cover solve all the violations sets in D. By Lemma 5, no new inconsistencies are added by applying the local repairs. Therefore, $D(\mathscr{C})$ is a repair of D wrt IC. We need to prove that $D(\mathscr{C})$ is an LS-repair. By contradiction, assume it is not. In this case, there exists an LS-repair D' for which $\Delta(D,D') < \Delta(D,D(\mathscr{C}))$.

Let $\mathscr{C}(D,D')$ be the cover defined as in Lemma 6. It holds $D(\mathscr{C}(D,D')) = D'$. Also, for different local repairs t' and t'' such that $S(t,t'), S(t,t'') \in \mathscr{C}(D,D')$, we have $adiff(t,t') \neq adiff(t,t'')$. Since all the local repairs modify different values, $w(\mathscr{C}(D,D')) = \Delta(D,D')$. But $\Delta(D,D') < \Delta(D,D(\mathscr{C})) = w(\mathscr{C})$. Therefore, $w(\mathscr{C}(D,D')) < w(\mathscr{C})$. We obtain that $\mathscr{C}$ is not a minimum cover, and we have a contradiction.

Now we prove the second statement, i.e. that for every LS-repair D' there is a minimum cover $\mathscr{C}$ such that $D(\mathscr{C}) = D'$. Let $\mathscr{C}(D,D')$ be defined as in Lemma 6. By Lemma 6, $\mathscr{C}(D,D')$ is a cover of the MWSCP for D and IC. We need to prove that $\mathscr{C}(D,D')$ is minimum. Let us assume, by contradiction, that there exists a minimum cover $\mathscr{C}'$ such that $w(\mathscr{C}') < w(\mathscr{C}(D,D'))$. By the first claim in this theorem, $D(\mathscr{C}')$ is an LS-repair. Since $\mathscr{C}'$ is a minimum cover, by Lemma 4, no two local repairs of a tuple modify the same attribute. As a consequence, $w(\mathscr{C}') = \Delta(D,D(\mathscr{C}'))$.

By construction, $\mathscr{C}(D,D')$ is such that no two local repairs of a tuple modify the same attribute. Thus, it holds that $w(\mathscr{C}(D,D')) = \Delta(D,D(\mathscr{C}))$. This implies that $\Delta(D,D(\mathscr{C}')) < \Delta(D,D(\mathscr{C}(D,D'))) = \Delta(D,D')$, which contradicts the fact that D' is an LS-repair. □

Example 21 (*Example 16 continued*). From the minimum cover $\mathscr{C}_1 = \{S(t_1,t_1''), S(t_2,\ t_2')\}$, we get $t_1^\star = t_1'' = Client(1,\mathbf{18},52)$, and $t_2^\star = t_2' = Client(2,16,\mathbf{50})$, and therefore, $D(\mathscr{C}_1)$ is instance D'' in Example 5. On the other hand, for the minimum cover $\mathscr{C}_2 = \{S(t_1,t_1'), S(t_2,t_2'), S(t_4,t_4'), S(t_5,t_5')\}$, the LS-repair $D(\mathscr{C}_1)$ coincides with instance D' in Example 5.

Proposition 5. *The transformation of an instance of DROP(IC) into the instance of MWSCP, and the construction of database instance $D(\mathscr{C})$ from a cover $\mathscr{C}$ for $(U,\mathscr{S},w)$ can be done both in polynomial time in the size of D.*

Proof. It remains to verify that $(U,\mathscr{S},w)$ can be computed in polynomial time in the size of D. Let n be the number of tuples in D. Notice that if m_i is the number of database atoms in $ic_i \in IC$, and m is the maximum value of the m_i, there are at most n^{m_i} violation sets associated to $ic_i \in IC$, each of them having between 1 and m tuples. Therefore, the size of U is $O(n^m \times |IC|)$. The size of each $S(t,t')$ is bounded by the size of U.

The size of $\mathscr{S}$, i.e. the number of sets $S(t,t')$, is polynomially bounded by the size of D. In fact, there is one $S(t,t')$ for each local repair of t, and each tuple may have no more than $|\mathscr{F}| \times |IC|$ local repairs, where $\mathscr{F}$ is the set of fixable attributes. Therefore, $|\mathscr{S}|$ is $O(|\mathscr{F}| \times |IC| \times |D|)$.

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 analysis similar to the one in Lemma 1).

With respect to $D(\mathscr{C})$, since $\mathscr{C} \subseteq \mathscr{S}$, $\mathscr{C}$ is polynomially bounded by the size of D. The generation of $T^\star$ and the replacements in D are easy. Therefore, the construction of $D(\mathscr{C})$ can be done in polynomial time on $|D|$. □

We have established that the transformation of *DROP* into *MWSCP* is an L-reduction [9].

If we apply this reduction to D and IC when IC is non-local, the instance $D(\mathscr{C})$ for a cover $\mathscr{C}$ can still be constructed, as above. However, it may not satisfy IC, because repairing single inconsistent tuples through local repairs solves only the initial inconsistencies, but new inconsistencies can be introduced. This is the case in Example 6, which has a non-local set of denials.

5.3. Approximate LS-repairs via approximate minimum covers

Now that we have transformed the database repair optimization problem into a weighted minimum set cover problem, we can apply approximation algorithms for the latter to approximate the former. Any cover $\mathscr{C}$, even if not minimum, will generate a database $D(\hat{\mathscr{C}})$ that is a repair of D. The better the approximation, the closer will the repair be to an LS-repair. For example, using a greedy algorithm, *MWSCP* can be approximated within a factor $\log(N)$, where N is the size of the underlying set U [16]. The approximation algorithm returns not only an approximation $\hat{w}$ to the optimal weight w^o, but also a—not necessarily optimal—cover $\hat{\mathscr{C}}$ for problem $(U,\mathscr{S},w)$. As in Definition 10, the cover $\hat{\mathscr{C}}$ can be used to generate a repair $D(\hat{\mathscr{C}})$ for D that may not be LS-minimum.

Example 22 (*Examples 5 and 21 continued*). We show how to compute a solution to this particular instance of *DROP* using the greedy approximation algorithm for *MWSCP* presented in [16]. We start with $\hat{\mathscr{C}}{:=}\emptyset$, $S(t_i,t_i')^0{:=}S(t_i,t_i')$ for each $S(t_i,t_i')\in\mathscr{S}$. Then, we add to $\mathscr{C}$ the set $S(t_i,t_i')$ such that $S(t_i,t_i')^0$ has the maximum *contribution ratio* $|S(t_i,t_i')^0|/w(S(t_i,t_i')^0)$. In this case, the alternatives are:

$$|S(t_1,t_1')|/w(S(t_1,t_1'))=1/4,$$
$$|S(t_1,t_1'')|/w(S(t_1,t_1''))=3/9,$$
$$|S(t_2,t_2')|/w(S(t_2,t_2'))=1,$$
$$|S(t_4,t_4')|/w(S(t_4,t_4'))=1/4,$$
$$|S(t_5,t_5')|/w(S(t_5,t_5'))=1.$$

The ratio is maximum for $S(t_2,t_2')$ and $S(t_5,t_5')$, so we can add any of them to $\hat{\mathscr{C}}$. If we choose the first, we get $\hat{\mathscr{C}}=\{S(t_2,t_2')\}$. Now we compute the new sets $S(t_i,t_i')^1{:=}S(t_i,t_i')^0\backslash S(t_2,t_2')^0$, and choose again an $S(t_i,t_i')$ for $\hat{\mathscr{C}}$ such that $S(t_i,t_i')^1$ maximizes the contribution ratio. Now $S(t_5,t_5')$ is added to $\hat{\mathscr{C}}$, because $S(t_5,t_5')^1$ gives the maximum.

We repeat this process until we get all the elements of U covered, i.e. all the sets $S(t_i,t_i')^k$ become empty at some iteration point k. We finally obtain $\hat{\mathscr{C}}=\{S(t_2,t_2'),S(t_5,t_5'),S(t_1,t_1'),S(t_4,t_4')\}$. In this case, $\hat{\mathscr{C}}$ is an optimal cover, and therefore, $D(\hat{\mathscr{C}})$ is an LS-repair, namely D' in Example 5.

Since we are using a cover that is not necessarily minimum, Lemma 4 may not hold. This means that, for a tuple t, there might be two local repairs in the cover that modify the value of the same attribute; and $t^\star$ cannot be computed as in Definition 10. This situation is shown in the following example.

Example 23 (*Example 18 continued*). The greedy algorithm in [16] returns the non-optimal cover $\hat{\mathscr{C}}=\{S(t,t_1),S(t,t_2),S(t,t_3)\}$. Here $t_1=R(1,\mathbf{3},1,8)$, $t_2=R(1,\mathbf{5},1,8)$ and $t_3=R(1,6,\mathbf{4},8)$ are all local repairs for the same tuple t, but now the first two of them modify its second attribute B. The tuples t_1 and t_2 solve different set of inconsistencies (violation sets); and those changes have to be made in order to solve them. Picking up randomly one of the two local repairs may leave the other set of inconsistencies unsolved.

However, in this example, B always appears in the denial constraints in comparisons of the form $B>c$ for a certain constant c.[9] Furthermore, the values for the attributes other than B are the same in t_1 and t_2. Therefore, we can construct the combined local repair by taking only the smallest value assigned to B in a local repair, namely 3 (the smallest constant c appearing in the comparisons $B>c$ above). Thus, the combined local repair would be in this case $t^\star=R(1,\mathbf{3},\mathbf{4},8)$, which solves all the violation sets.

[9]Recall that since the constraints are local, a fixable attribute is either used with $<$ or $>$ in IC but never both.

We need to modify Definition 10 to consider not only minimum covers, but any cover that might be returned by an approximation algorithm.

Definition 12. Let $\hat{\mathscr{C}}$ be a (not necessarily minimum) cover for instance $(U,\mathscr{S},w)$ of the *MWSCP* associated to D and IC. (a) For each tuple $t\in D$, and its local repairs $t_1,\ldots,t_n$ for which $S(t,t_i)\in\mathscr{C}$ holds, the *combined local repair* $t^\star$ is defined by the following conditions: (a1) $t^\star[A]=t_i[A]$ if there exists an i in $[1,n]$ with $A=\mathit{adiff}(t,t_i)$; and there is no other j in $[1,n]$ with $A=\mathit{adiff}(t,t_j)$ and $t_j[A]<t_i[A]$ ($t_j[A]>t_i[A]$) if A is compared with $>$ (resp. $<$) in IC. (a2) $t^\star[A]=t[A]$ for other attributes A that are not involved in (a1).

(b) $T^\star$ and instance $D(\hat{\mathscr{C}})$ associated to $\hat{\mathscr{C}}$ are defined as in Definition 10.

This definition coincides with Definition 10 if $\hat{\mathscr{C}}$ is a minimum cover. Indeed, if we have a minimum cover, Lemma 4 holds, and for each tuple there is a unique local repair that modifies a specific attribute.

Example 24 (*Example 23 continued*). Applying Definition 12 to the non-minimum cover $\hat{\mathscr{C}}=\{S(t,t_1),S(t,t_2),\ S(t,t_3)\}$, we obtain $t^\star=R(1,\mathbf{3},\mathbf{4},8)$, and then $D(\hat{\mathscr{C}})=\{R(1,\mathbf{3},\mathbf{4},8)\}$, which is an LS-repair of D wrt IC.

This example shows that even when the approximation algorithm returns a non-minimum cover, the repair associated to it may be minimum, i.e. an LS-repair. However, non-minimum covers may also lead to non-minimum repairs.

Example 25. Consider a schema with predicates $P(A,B,C),Q(D,E,F,G)$, and $K_P=\{A\}$, $K_Q=\{D\}$, and $\mathscr{F}=\{B,C,E,F\}$. The instance D below is inconsistent wrt the local ICs $ic_1:\bar{\forall}\neg(P(x,y,z),Q(x,w,v,u),y>7,w<7)$, $ic_2:\bar{\forall}\neg(P(x,y,z),Q(x,w,v,u),y>9,\ w<6)$, $ic_3:\bar{\forall}\neg(P(x,y,z),y>7,z>2)$, and $ic_4:\bar{\forall}\neg(P(x,y,z),Q(u,w,v,x),\ y>9,w>2)$.

D:

P	A	B	C		Q	D	E	F	G	
	a	11	4	t_1		a	4	5	f	t_3
	b	5	7	t_2		c	4	8	a	t_4

The violation sets are $(\{t_1,t_3\},ic_1)$, $(\{t_1,t_3\},ic_2)$, $(\{t_1\},ic_3)$ and $(\{t_1,\ t_4\},ic_4)$. The local repairs are: $t_1':P(a,\mathbf{7},4)$, $t_1'':P(a,\mathbf{9},4)$, $t_3':P(a,\mathbf{7},5,f)$, $t_3'':P(a,\mathbf{6},5,f)$, and $t_4':P(c,\mathbf{2},8,a)$. The instance of the MWSCP is

Set	$S(t_1,t_1')$	$S(t_1,t_1'')$	$S(t_3,t_3')$	$S(t_3,t_3'')$	$S(t_4,t_4')$
Weight	16	4	9	4	4
$(\{t_1,t_3\},ic_1)$	1	0	1	0	0
$(\{t_1,t_3\},ic_2)$	1	1	1	0	0
$(\{t_1\},ic_3)$	1	0	0	1	0
$(\{t_1,t_4\},ic_4)$	1	0	0	1	1

The only minimum cover for this problem is $\mathscr{C}=\{S(t_1,t_1')\}$, with weight 16. As expected, the associated repair $D(\mathscr{C})$ also has distance 16 to D. However, a cover obtained by the greedy algorithm in [16] is $\hat{\mathscr{C}}=\{S(t_1,t_1''),S(t_3,t_3'),S(t_3,t_3''),S(t_4,t_4')\}$, with weight 21. The repair $D(\hat{\mathscr{C}})$ is

P	A	B	C		Q	D	E	F	G	
	a	**9**	4	t_1''		a	**7**	5	f	t_3'
	b	5	7	t_2		c	**2**	8	a	t_4'

This repair is not an LS-repair since the distance to D is 17 instead of 16.

Notice that, as illustrated in the previous example, for a cover $\mathscr{C}$, it holds $\Delta(D,D(\mathscr{C}))\leqslant w(\mathscr{C})$, with equality for a minimum cover. This is because, to obtain $D(\mathscr{C})$, we may eliminate elements from $\mathscr{C}$.

Proposition 6. *Given an instance D and a set IC of local ICs, the instance $D(\hat{\mathscr{C}})$, obtained from the cover $\hat{\mathscr{C}}$ returned by the approximation algorithm, is a repair. It also holds $\Delta(D,D(\hat{\mathscr{C}}))\leqslant \log(N)\times\Delta(D,D')$, where D' is any LS-repair of D wrt IC, and N is the number of violation sets for D wrt IC.*

Proof. Since $\hat{\mathscr{C}}$ is a cover, the local repairs that it represents solve all the violations sets in D. Since we are dealing with local ICs, the set of updates defined by $T^\star$ still solves all the inconsistencies. By Lemma 5, no new inconsistencies are added by applying the local repairs. Therefore, $D(\hat{\mathscr{C}})$ is a repair of D wrt IC.

We need to prove that $\Delta(D,D(\hat{\mathscr{C}}))\leqslant \log(N)\times\Delta(D,D')$. By definition of the square distance, it holds $\Delta(D,D(\hat{\mathscr{C}}))=\sum_{t\in D}\Delta(\{t\},\{t^\star\})$. Let $\{t_1,\ldots,t_n\}$ be the set of local repairs of t such that $\{S(t,t_1),\ldots,S(t,t_n)\}\subseteq\hat{\mathscr{C}}$. When computing $t^\star$ for t, some of the $S(t,t_i)$ in $\hat{\mathscr{C}}$ may not have been used. Assume that t^* is built using the local repairs $t_{i_1},\ldots,t_{i_k}$, with $\{i_1,\ldots,i_k\}\subseteq\{1,\ldots,n\}$. Thus, it holds $\Delta(\{t\},\{t^\star\})=\sum_{j=1}^{k}\ \Delta(\{t\},\{t_{i_j}\})\leqslant\sum_{i=1}^{n}\Delta(\{t\},\{t_i\})$.In consequence, $\sum_{t\in D}\Delta(\{t\},\{t^\star\})\leqslant\sum_{S(t,t_i)\in\hat{\mathscr{C}}}\Delta(\{t\},\{t_i\})$. Finally, we obtain $\Delta(D,D(\hat{\mathscr{C}}))\leqslant\sum_{S(t,t_i)\in\hat{\mathscr{C}}}\ \Delta(\{t\},\{t_i\})=\sum_{S(t,t_i)\in\hat{\mathscr{C}}}\ w(S(t,t_i))=\hat{w}\leqslant\log(N)\times w^o=\log(N)\times\Delta(D,D')$, for every LS-repair D' of D. $\square$

We have obtained that, for any set IC of local denials, there is a polynomial time approximation algorithm that solves $DROP(IC)$ within an $O(\log(N))$ factor, where N is the number of violation sets for D wrt IC. As mentioned before, this number N is polynomially bounded by $|D|$ (cf. Proposition 5). N may be small if the number of inconsistencies or the number of database atoms in the ICs are small, which is likely the case in real applications.

However, we can get an even better approximation via a cover $\hat{\mathscr{C}}$ obtained with an approximation algorithm for a special case of the *MWSCP*: When the number of occurrences of an element of U in elements of $\mathscr{S}$ (its frequency) is bounded by a constant. For this case of the *MWSCP* there are approximations based on "linear relaxation" that provide a constant approximation factor [13, Chapter 3]. This is clearly the case in our application, being $m\times|\mathscr{F}|\times|IC|$ a constant bound (independent from $|D|$) on the *frequency* of each element of U, where m is the maximum number of database atoms in an IC.

Theorem 6. *There is a polynomial time approximation algorithm that, for a given instance D and a set IC of local ICs, returns a repair $D(\hat{\mathscr{C}})$ of D wrt IC, such that $\Delta(D,D(\hat{\mathscr{C}}))\leqslant c\times\Delta(D,D')$, where c is a constant and D' is any LS-repair of D.*

6. One atom denials and conjunctive queries

In this section we concentrate on the common case of *one database atom denials* (1AD), i.e. of the form $\forall\neg(A,B)$, where atom A has a predicate in $\mathscr{R}$, and B is a conjunction of built-in atoms. One atom denials are used in practice. For example, they capture range constraints. They can also be used as constraints on census data, which is usually represented as a single relation [3].

It is not difficult to produce examples of instances that have exponentially many LS-repairs wrt 1ADs.

Thus, CQA is not necessarily easier under 1ADs. Actually, we will see below that some decision problems around CQA are *NP*-hard.

For 1ADs, we can identify tractable cases for *CQA* under LS-repairs by reduction to *CQA* for (tuple and set-theoretic) repairs of the form introduced in [2] for key constraints. This is because each violation set (cf. Definition 7) contains one tuple, maybe with several local repairs, but all sharing the same rigid values. So, now the problem consists in choosing one from different tuples with the same rigid values (cf. proof of Theorem 7 below). The transformation preserves consistent answers to both ground and open queries.

The "classic"-tuple and set oriented—CQA problem as introduced in [2] has been studied in detail for key dependencies in [14,17]. In particular, for tractability of *CQA* in our setting, we can use results and algorithms obtained in [17] for the classic framework.

The *join graph* $\mathcal{G}(Q)$ [17] of a conjunctive query without built-ins Q is a directed graph whose vertices are the database atoms in Q. There is an edge 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.

For a conjunctive query without repeated database predicates and without built-ins Q, we write $Q \in \mathcal{C}_{Forest}$ if $\mathcal{G}(Q)$ is a forest and every non-key to key join of Q is full i.e. involves the whole key. Classic *CQA* wrt key constraints is tractable for queries in $\mathcal{C}_{Forest}$ [17].

Theorem 7. *For every fixed set of 1ADs and query in C_{Forest}, CQA under LS-repairs is in PTIME.*

Proof. Based on the tractability results in [17], it suffices to show that the LS-repairs for a database D are in one-to-one and polynomial time correspondence with the classic repairs based on tuple deletions [2,14] for a database D' wrt a set of key dependencies.

Since we have 1ADs, the violation sets will have a single element. Thus, for an inconsistent tuple t wrt a constraint $ic \in IC$, it holds $(\{t\}, ic) \in \mathcal{I}(D, IC)$. Since all the violation sets are independent, in order to compute an LS-repair for D, we have to generate independently all the local repairs t' for all inconsistent tuples t such that there exists and $ic \in IC$ with $(\{t\}, ic) \in \mathcal{I}(D, IC)$.

Those local repairs can be found by considering all the *candidate* repairs (not necessarily LS-minimum) that can be obtained by replacing, in each tuple, the flexible attributes that appear in the ICs by all the values in the intervals around the borders and around the values in D (cf. Lemma 1). Then, one can check which of candidate repairs satisfy IC. Finally, those that minimize $\Delta(\{t\}, \{t'\})$ are chosen. The number of repair candidates per tuple in the database is $O((|D| \times |\mathcal{A}|^2 + |Borders(IC)| \times |\mathcal{A}|)^{|\mathcal{F}|})$, where $\mathcal{A}$ is the set of attributes and $\mathcal{F}$ is the set of fixable attributes. Thus, the total number of repair candidates is $O(|D|(|D| \times |\mathcal{A}|^2 + |Borders(IC)| \times |\mathcal{A}|)^{|\mathcal{F}|})$, which is polynomial in the size of D.

Let us now define a database D' consisting of the consistent tuples in D, together with all the local repairs of the inconsistent tuples. By construction, D and D' share the same rigid values. Since each inconsistent tuple in D may have more than one local repair, D' may become inconsistent wrt its key constraints. Each classic repair of D', i.e. obtained by tuple deletions, will choose one local repair from D' for each inconsistent tuple t of D, and therefore will determine an LS-repair of D wrt IC. Conversely, every LS-repair can be obtained in this way. □

For queries Q returning numerical values only, e.g. scalar aggregate queries, which is common in our framework, it is natural to use a *range semantics* for *CQA* [11]. In this case, the consistent answer to Q is the pair consisting of the *max–min* and *min–max* answers, i.e. the infimum and supremum, resp., of the set of answers to Q obtained from LS-repairs. In other words, the consistent answer to a numerical query Q is the shortest interval [*max–min*, *min–max*], such that for every LS-repair D', $Q(D') \in [max{-}min, min{-}max]$.

We can see that the *max–min* and *min–max* answers to a query are the minimum and the maximum answers, resp., considering all LS-repairs. So, finding these values becomes a minimization and a maximization problem, resp., over the class of LS-repairs. Correspondingly, the *decision problems of CQA under the range semantics* consist in determining if a numerical query Q has its answer: (a) $\leqslant k_1$ in some LS-repair (the *max–min* case); or (b) $\geqslant k_2$ for some LS-repair (the *min–max* case).

Our next result exhibits, for each of the common aggregate functions, a set of denials and a scalar aggregate conjunctive query for which at least one of these two decision problems of CQA becomes *NP*-complete.

Theorem 8. *For each of the aggregation functions sum, count, count distinct, and average, there is a fixed set of 1ADs and a fixed aggregate acyclic conjunctive query with one occurrence of the function, such that CQA under the range semantics is NP-complete.*

For the proof of this theorem we need a preliminary result.

Lemma 7. *Consider a regular undirected graph* $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ *of degree* 3 (*i.e. all the vertices have degree* 3), *and a function F from sets of vertices S to nonnegative integers defined as follows*:

(1) *For* $S \subseteq \mathscr{V}$, *and* $v \in \mathscr{V}$, $F^l(S, v) := |T(S, v)|^3$, *where*

$$T(S,v) := \begin{cases} \{v' | v' \in (\mathscr{V} \backslash S) \text{ and } \{v, v'\} \in \mathscr{E}\}, & v \in S, \\ \emptyset, & v \notin S, \end{cases}$$

(2) $F(S) := \sum_{v \in S} F^l(S, v)$.

The maximum value of $F(S)$ *over all possible sets* $S \subseteq \mathscr{V}$ *is* $(3^3 \times |I|)$, *for I a maximal* (*wrt set inclusion*) *independent set.*

Proof. Let us first assume that S is an independent set, not necessarily maximal. In this case the value $F(S)$ is $3^3 \times |S|$, because each element $v \in S$ is connected to three vertices in $\mathscr{V} \backslash S$. Then, among independent sets, the maximum value for $F(S)$ is $3^3 \times m$, where m is the maximum cardinality of an independent set.

For $S \subseteq \mathscr{V}$, let $\mathscr{G}[S]$ denote the subgraph $(S, \mathscr{E}_S)$, where $\mathscr{E}_S$ are all the edges $\{v, v'\} \in \mathscr{E}$ such that $v, v' \in S$. Now, if S is not an independent set, there exists a maximum independent set I_S of $\mathscr{G}[S]$. Every $v \in (\mathscr{V} \backslash S)$ is adjacent to at least one vertex in I_S, otherwise $I_S \cup \{v\}$ would be an independent set contained in S which is a proper extension of I_S, contradicting our choice of I_S. Now, define $F_{ext}(S, v) = (F^l(S, v) + \sum_{\{v,v'\} \in \mathscr{E}} F^l(S, v'))$. Since every edge $v' \in (S \backslash I_S)$ is adjacent to I_S, it is easy to see that

$$F(S) \leqslant \sum_{v \in I} F_{ext}(S, v). \tag{2}$$

We want to prove that $F(S) \leqslant F(I_S)$. This, combined with Eq. (2), shows that it suffices to prove that $\sum_{v \in I_S} F_{ext}(S, v) \leqslant F(I_S)$. Since $F(I_S) = \sum_{v \in I_S} F^l(I_S, v)$, we need to prove that $\sum_{v \in I_S} F_{ext}(S, v) \leqslant \sum_{v \in I_S} F^l(I_S, v)$. It is sufficient to prove that $F_{ext}(S, v) \leqslant F^l(I_S, v)$ is true for every $v \in I_S$. For $v \in I_S$ and $S' = (S \backslash I_S)$, we have the following cases:

(1) If v is adjacent to one vertex in S', then $F_{ext}(S, v) \leqslant 2^3 + 2^3$, and $F^l(I_S, v) = 3^3$. In consequence, $F_{ext}(S, v) \leqslant (F^l(I_S, v) - 11)$.
(2) If v is adjacent to two vertices in S', analogously to (1), we get $F_{ext}(S, v) \leqslant (F^l(I_S, v) - 10)$.
(3) If v is adjacent to three vertices in S', analogously to (1), we get $F_{ext}(S, v) \leqslant (F^l(I_S, v) - 3)$.

Thus, we have proved that $F_{ext}(S, v) \leqslant F^l(I_S, v)$, and therefore, that $F(S) \leqslant F(I_S)$. We also know that, since I_S is an independent set, that $F(S) \leqslant F(I_S) \leqslant 3^3 \times m$. □

Proof of Theorem 8. In all the cases, membership to *NP* is a consequence of the existence of a polynomially bounded and polynomial-time verifiable certificate, as established in Theorem 1. Now we consider hardness.

(a) For *sum*: By reduction from the *NP*-hard problem *Independent Set for Cubic Planar Graphs* [18], where the vertices of the graph have all degree 3.

Given an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ of degree 3, and a lower bound k for the size for a maximum independent set, we create a predicate $Ver(V, C_1, C_2)$, where the key V takes vertices as values, and C_1, C_2 are fixable and may take values 0 or 1, but are all equal to 0 in the initial instance D. This relation is subject to the denial ic: $\forall v, c_1, c_2 \neg(Ver(v, c_1, c_2), c_1 < 1, c_2 < 1)$. D is inconsistent wrt this constraint, and, in any of its LS-repairs, each vertex v will have associated a tuple $Ver(v, 1, 0)$ or $Ver(v, 0, 1)$, but not both.

Each LS-repair D' of the database defines a partition of $\mathscr{V}$ into two subsets: S, with those v with $Ver(v, 1, 0) \in D'$; and S', with those v with $Ver(v, 0, 1) \in D'$. Clearly $S \cup S' = \mathscr{V}$ and $S \cap S' = \emptyset$. We also use predicate $Edge(V_1, V_2, W)$, with rigid attributes only, whose extension contains $Edge(v_1, v_2, 1)$ and $Edge(v_2, v_1, 1)$ for $\{v_1, v_2\} \in \mathscr{E}$. So, every vertex v appears exactly 3 times in each argument of *Edge*-tuples in D.

Consider the ground aggregate conjunctive query Q:

$$q(sum(w_0)) \leftarrow Ver(v_1, c_{11}, c_{12}),$$
$$c_{11} = 1, Edge(v_1, v_2, w_0), Ver(v_2, c_{21}, c_{22}),$$
$$c_{21} = 0, Edge(v_1, v_3, w_1), Ver(v_3, c_{31}, c_{32}), c_{31} = 0,$$
$$Edge(v_1, v_4, w_2), Ver(v_4, c_{41}, c_{42}), c_{41} = 0.$$

Given an LS-repair as a partition of a set of vertices into two subsets S and S', for each vertex $v \in S$, the body of Q will be satisfied m^3 times, where m is the number of vertices of S' that are adjacent to elements in S. This happens because the query has three pairs of predicates *Edge*, *Ver*, each predicate can be satisfied m times, and they are independent from each other (there are no equality, non-equality or predicates connecting them). In consequence, the whole body of the query will be satisfied m^3 times. Each satisfying assignment of the body of the query brings value 1 to the multiset (w_0) which is under the sum aggregation function. That is, query Q computes the nonnegative integer function F, such that $F(S)$ gives the sum of cubes of the number of vertices of $\mathscr{V} \backslash S$ that are adjacent to vertices in S. More precisely, $F(S) = Q(D')$, for $D' \in Rep(D, IC)$ and $S = \{v | Ver(v, 1, 0) \in D'\}$.

Since this function is nonnegative and its value is zero for $S = \emptyset$ and $S = \mathscr{V}$, we have that its minimum value in the repairs is zero. We are interested in the maximum value for Q in $Rep(D, IC)$, i.e. the *min–max answer* introduced in [11].

From Lemma 7, we have that the answer to query Q is at most $3^3 \times |I|$, with I a maximum independent set. In consequence, the min–max answer for Q is $3^3 \times m$, with m the cardinality of the maximum independent set. Thus, there is an independent set of size at least k iff min–max answer to $Q \geqslant 3^3 \times k$.

(b) For *count*: Basically the same proof as for (a) applies. Only the query has to be changed to:

$$\begin{aligned}&q(count) \leftarrow Ver(v_1, c_{11}, c_{12}),\\ &c_{11} = 1, Edge(v_1, v_2, w_0), Ver(v_2, c_{21}, c_{22}),\\ &c_{21} = 0, Edge(v_1, v_3, w_1), Ver(v_3, c_{31}, c_{32}), c_{31} = 0,\\ &Edge(v_1, v_4, w_2), Ver(v_4, c_{41}, c_{42}), c_{41} = 0, w_0 = 1.\end{aligned}$$

(c) For *count distinct*: By reduction from *MAXSAT*. Assume that an instance for *MAXSAT* is given. It consists of a set $\mathscr{P}$ of propositional variables, a collection $\mathscr{C}$ of clauses over $\mathscr{P}$, and a positive integer k. The question is whether at least k clauses can be satisfied simultaneously. The answer will be *Yes* exactly when a question of the form $countd \leqslant (k-1)$, with *countd* defined by an aggregate query over a database instance (both of them to be constructed below), gets answer *No* under the min–max semantics.

Define a predicate $Var(U, V_1, V_2)$, being the first attribute the key, and the second and third are fixable (the denial below and the minimality condition will make them take values 0 or 1). The initial database contains $Var(u, 0, 0)$ for every $u \in \mathscr{P}$. Another predicate, $Clause(u, c, s)$, has no fixable attributes. Its extension contains $Clause(u, c, s)$ for every occurrence of variable $u \in \mathscr{P}$ in a clause $c \in \mathscr{C}$, where s an assignment for u satisfying clause c. The IC is $\forall u, v_1, v_2 \neg(Var(u, v_1, v_2), v_1 < 1, v_2 < 1)$. The acyclic query is $q(countd(c)) \leftarrow Var(u, v_1, v_2), Clause(u, c, s), v_1 = s$, where *countd* denotes the "count distinct" aggregation function. Its answer tells us how many clauses are satisfied in a given LS-repair. The *max* value taken on a LS-repair, i.e. the min–max answer, will be the maximum number of clauses which may be satisfied for *MAXSAT*.

(d) For *average*: By reduction from 3-*SAT*. We use the same predicate $Var(U, V_1, V_2)$ and IC as in (c). Now, we encode clauses as tuples in the extension of a predicate $Clause(Val, Var_1, Val_1, Var_2, Val_2, Var_3, Val_3)$, which has no fixable attributes. The extension contains tuples $Clause(val, var_1, val_1, var_2, val_2, var_3)$, where var_1, var_2, var_3 are the propositional variables in the clause (in any order), val_1, val_2, val_3 are all the possible combinations of truth assignments to variables (at most eight combinations per clause); and *val* is the corresponding truth value taken by the clause (0 or 1) given the values val_i. Now, the acyclic query

$$\begin{aligned}&q(avg(v))\\ &\quad \leftarrow Clause(v, u_1, v_1, u_2, v_2, u_3, v_3),\\ &\qquad Var(u_1, v_1, v_1'), Var(u_2, v_2, v_2'), Var(u_3, v_3, v_3')\end{aligned}$$

takes a maximum value 1 in an LS-repair, i.e. the min–max answer to q is 1, iff the formula is satisfiable. □

Notice that for the four aggregation functions one 1AD suffices (plus the 1ADs that force numerical values not to be less that 0). For *sum* and *count* we use a reduction from the *Independent Set Problem* with bounded degree 3 [18]. The general *Independent Set Problem* has bad approximation properties [13, Chapter 10]. The *Bounded Degree Independent Set* has efficient approximations within a constant factor that depends on the degree [19].

Theorem 9. *For any set of 1ADs and conjunctive query with sum over a non-negative attribute, there is a polynomial time approximation algorithm with constant factor for determining the min–max answer for CQA.*

The factor in this theorem depends upon the ICs and the query, but not on the size of the database. The acyclicity of the query is not required. The algorithm is based on a reduction of our problem to satisfying a subsystem with maximum weight of a system of weighted algebraic equations over the Galois field with two elements $GF[2]$ (a generalization of problems in [20,21]). For the latter problem, a polynomial time approximation similar to the one for *MAXSAT* can be given [21]. The long proof of this theorem is given in Appendix A in [22].

7. Extensions

7.1. Dependencies between attributes

The notion of LS-repair that we introduced can be seen as a first approach to the problem of defining and computing a semantically correct data set that is close to the one at hand. Thinking of census-like applications, the idea that key constraints are satisfied makes sense. Data forms usually come partially filled out, with the data for the identification fields already entered. It is the data that is entered *in situ* the one that is subject to errors. Considering the possible violation of the key constraints in this picture would require much more research. Our results rely on the hardness of the key constraints.

We are also making the reasonable assumption that the dependencies between fixable attributes are captured by the denial constraints, but the errors that one can make when entering the individual attribute values are independent from each other. A more sophisticated model could consider some sort of stochastic dependency between the errors made in groups of attribute values. This dependency should be captured by the distance function. This is an interesting venue to explore that has to be left for future research.

7.2. Minimum distribution-variation repairs

Repairing a database under minimization of the square distance to the original database may not preserve the statistical or aggregate properties of the original data, which in some applications could be relevant, like in census data. Given that there may be several LS-repairs, we may prefer those that preserve the data distribution.

As a first step in this direction, one could choose those LS-repairs such that, for each fixable attribute, the frequency of values in the repair stay as close as possible to the frequency of values in the original database. This is one way of capturing the preservation of the data distribution. This choice assumes that the different attributes are stochastically independent. Without attempting to develop this direction in full, we briefly investigate in the following a first possible approach. A deeper analysis of this class of LS-repairs under the independence assumption, and also the study of preservation of statistical properties when attributes are stochastically correlated are both material for future research.

Definition 13. Let D and D' be instances over the schema $\Sigma = (\mathscr{U}, \mathscr{R}, \mathscr{B}, \mathscr{A})$, and $R \in \mathscr{R}$. (a) The *distribution distance between D and D' wrt attribute A of R* is $\Delta_d^{R.A}(D, D') = \sum_{a \in Dom(A)} (count(a, R.A, D) - count(a, R.A, D'))^2$, where $count(a, R.A, D)$ gives the number of occurrences of value a in A of R in instance D.

(b) The *distribution distance* $\Delta_d(D, D')$ between D and D' is the maximum of the distribution distances over all relations and their attributes.

(c) Given set of ICs IC, D' is a *minimum distribution variation repair* (MDV-repair) of D wrt IC if D' is an LS-repair that also minimizes $\Delta_d(D, D')$.

Example 26. Consider the IC $\forall N, E, S \neg (Emp(N, E, S), E < 5, S > 5)$, which requires that no employee with experience shorter than 5 years gets a salary higher than 5 thousand. The inconsistent instance $D = \{Emp(Ann, 4, 6), Emp(Bill, 3, 7), Emp(Chris, 2, 2), Emp(Dan, 6, 6)\}$ has the following LS-repairs:

$$D_1 = \{Emp(Ann, 4, 5), Emp(Bill, 3, 5), Emp(Chris, 2, 2), Emp(Dan, 6, 6)\},$$

$$D_2 = \{Emp(Ann, 4, 5), Emp(Bill, 5, 7), Emp(Chris, 2, 2), Emp(Dan, 6, 6)\},$$

$$D_3 = \{Emp(Ann, 5, 6), Emp(Bill, 3, 5), Emp(Chris, 2, 2), Emp(Dan, 6, 6)\},$$

$$D_4 = \{Emp(Ann, 5, 6), Emp(Bill, 5, 7), Emp(Chris, 2, 2), Emp(Dan, 6, 6)\}.$$

The distance is $\Delta(D, D_i) = 1^2 + 2^2 = 5$ for $i = 1, 2, 3, 4$.

$\Delta_d(D, D_1) = Max\{\Delta_d^N(D, D_1), \Delta_d^E(D, D_1), \Delta_d^S(D, D_1)\}$, with $\Delta_d^N(D, D_1) = 0$, $\Delta_d^E(D, D_1) = 0$, and $\Delta_d^S(D, D_1) = (count(5, S, D) - count(5, S, D_1))^2 + (count(6, S, D) - count(6, S, D_1))^2 + (count(7, S, D) - count(7, S, D_1))^2 = 2^2 + 1^2 + 1^2 = 6$. Thus, $\Delta_d(D, D_1) = 6$.

$\Delta_d(D, D_2) = Max\{\Delta_d^N(D, D_2), \Delta_d^E(D, D_2), \Delta_d^S(D, D_2)\}$, with $\Delta_d^N(D, D_1) = 0$, $\Delta_d^E(D, D_2) = (count(3, S, D) - count(3, S, D_2))^2 + (count(5, S, D) - count(5, S, D_2))^2 = 1^2 + 1^2$, and $\Delta_d^S(D, D_2) = (count(5, S) - count(5, S, D_2))^2 + (count(6, S, D) - count(6, S, D_2))^2 = 1^2 + 1^2$. Thus, $\Delta_d(D, D_2) = 2$.

This shows that repair D_1 has a bigger impact over the distribution than D_2. Then, if we want to keep the statistical properties of the database we will prefer D_2 better than D_1. Analogously, we can obtain $\Delta_d(D, D_3) = 2$ and $\Delta_d(D, D_4) = 6$. Then, $\Delta_d(D, D_2) = \Delta_d(D, D_3) < \Delta_d(D, D_1) = \Delta_d(D, D_4)$, and the MDV-repairs are D_2 and D_3.

From Theorem 2 and the fact that for a database there is an LS-repair if and only if there is a MDV-repair, we obtain

Proposition 7. *The problem of existence of MDV-repairs under linear constraints is NP-complete.*

7.3. Aggregation constraints

We may consider aggregation constraints (ACs) [23] expressed in terms of aggregation functions, like *sum*, *count*, *average*. It is natural and common to use those functions when processing numerical data.

Filtering ACs impose conditions on the tuples over which aggregation is applied, e.g. $sum(A_1 : A_2 = 3) > 5$ contains a sum over A_1 of tuples with $A_2 = 3$, and checks if this sum is greater that 5. *Multi-attribute* ACs allow for arithmetical combinations of attributes as arguments for *sum*, e.g. $sum(A_1 + A_2) > 5$ and $sum(A_1 \times A_2) > 100$. If an AC has attributes from more than one predicate, it is *multi-relation*, e.g. $sum_{R_1}(A_1) = sum_{R_2}(A_1)$, otherwise it is *single-relation*.

Having aggregation constrains together with denial constrains has an impact on the class of possible repairs. For example, consider the relational predicate $R(A, B)$, with key A and fixable attribute B. If we have $IC = \{\forall x, y \neg(R(x, y), y > 1), sum(B) = 10\}$, any database with less than 10 tuples has no repairs.

It is not difficult to see that for a fixed set IC of ICs containing denials and aggregation constraints, $NE(IC)$ is decidable. Here we will investigate the decision $NE = \{(D, IC) | Rep(D, IC) \neq \emptyset\}$, of existence of LS-repairs, whose instances in this case consist both of a database instance and a set of constraints (and a corresponding schema). We consider the variant of CQA where ICs are also part of the instances.

Theorem 10. *Under extended linear denials and complex, filtering, multi-attribute, single-relation, aggregation constraints, the problems NE of existence of LS-repairs, and CQA under the skeptical semantics are undecidable.*

Proof (*sketch*). Hilbert's 10th problem on existence of integer solutions to diophantine equations can be reduced to NE. More precisely, given a diophantine equation, it is possible to construct a database D and a set of ICs IC such that the existence of an LS-repair for D wrt IC implies the existence of a solution to the equation, and vice versa. An example can be found in Appendix B in [22]. For CQA, apply Proposition 2. A more detailed sketch of the proof for NE and a representative example is given in Appendix B in [22]. $\square$

7.4. Other distances

Our results for the square distance rely basically on the additivity of the distance on the tuples of the database and its monotonicity on the absolute values of the differences between values for the same attribute. The property of polynomial-time computability of the distance function is also required. Any distance that satisfies these properties can be used instead of the L_2 distance, obtaining the same complexity results. In particular, all the results apply to the "city" distance (or L_1 distance) given by the sum of those absolute differences. Of course, when using the L_1 distance, we may get a different set of repairs for the same database. The approximation algorithm can also be used by computing the weights using the L_1 instead of the L_2 distance. Thus, the general complexity and approximability results still hold. For example, the optimization and implementation of the approximation in [24] of the algorithm for the *DROP* problem presented here uses the L_1 distance, without any essential changes wrt the treatment based on the L_2 distance.

The *edit distance* (ED) between two strings is the minimum number of substitutions, deletions and insertions of characters that are needed to transform one string into the other. The *hamming distance* between two strings of the same length is the number of positions that have different characters. The hamming distance is an upper bound for the edit distance. For example, $HD(234, 345) = 3$, but $ED(234, 345) = 2$. These distances are used in data editing, but they are more appropriate for strings of characters, and not for numerical data. Actually, our results would not apply to the edit distance or the hamming distance, because they do not monotonically increase over the absolute value of the difference between two attributes. For example, for the numbers 21, 30 and 31, $ED(21, 30) = 2 > ED(21, 31) = 1$. However, since $|21 - 31| > |21 - 30|$, the edit distance does not monotonically increase over the absolute value of the difference. The same example can be used for the hamming distance, because $HD(21, 30) = ED(21, 30)$ and $HD(21, 31) = ED(21, 31)$.

Another problem with the edit and hamming distance is that there are too many possible repairs to consider. For example, the database $D = \{P(a, 150)\}$, where the first attribute is the primary key and the second attribute is fixable, is inconsistent wrt $P(x, y) \rightarrow y \geqslant 200$. There exists only one LS-repair (under the quadratic distance): $D' = \{P(a, 200)\}$. If instead, we consider the edit distance, there are more than 40 repairs at distance one. For example: $D_1 = \{P(a, 250)\}$, $D_2 = \{P(a, 350)\}$, $D_3 = \{P(a, 1050)\}$, $D_4 = \{P(a, 1530)\}$, $D_5 = \{P(a, 1590)\}$, $D_6 = \{P(a, 6150)\}$, $D_7 = \{P(a, 1507)\}$, etc.

8. Conclusions

We have shown that fixing numerical values in databases poses many new computational challenges that had not been addressed before in the context of CQA. These problems are particularly relevant in census-like applications, where the problem of *statistical data editing* [25,26] is a common and difficult task. Also our concentration on aggregate queries is particularly relevant for this kind of statistical applications. In this paper we have just started to investigate some of the many problems that appear in this context, and several possible extensions deserve to be explored.

We concentrated on integer numerical values, which provide a useful and challenging domain. Considering real numbers in fixable attributes opens many new issues, requires different approaches; and must be left as a subject of future research. Some of the results presented here carry over to the case of real numbers. However, apart from the technical problems, the main complication is to come up with a right repair semantics in the presence of real numbers, in particular in comparisons of attributes. For example, if two attributes A, B take the same value, but a constraint prevents this from happening, it is not clear what new values have to be given to them in order to restore consistency. Most likely making them differ by an infinitesimal quantity would not make much sense in most of the applications. We could accept an epsilon of error in the distance, in such a way that if, for example, the distance of a repair is 5 and the distance to another repair is 5.001, we could take both of them as (minimum) LS-repairs.

What is essential about the numerical domain, in our case, the integers, is that we have a discrete linear order and a numerical distance function that is monotonic on the length of the interval between two arbitrary elements. In consequence, the framework established in this paper could be applied to qualitative attributes which have an implicit linear order given by the application. Also numerical distances, like the ones introduced here, could be applied to domains other than numerical if their elements can be naturally mapped to numbers.

The result we have presented for fixable attributes that are all equally relevant ($\alpha_A = 1$ in Definitions 1 and 2) should carry over without much difficulty to the general case of arbitrary weighted repairs. We have shown how to extend our approach in order to consider *minimum distribution variation* LS-repairs that keep, in some sense, the overall statistical properties of the database.

Other open problems refer to the identification of cases of polynomial complexity for linear denials with more that one database atom; approximation algorithms for the *DROP* for non-local cases; and approximations to *CQA* for other aggregate queries. More research on the impact of aggregation constraints on LS-repairs is needed.

For related work, we refer mainly to the literature on CQA. In [1] an earlier survey with abundant references can be found. More recent surveys can be found in [4,5]. Most of the research on CQA has been carried out appealing to a tuple oriented repair semantics, i.e. minimal repairs are obtained through tuple insertions or deletions. Under the set-theoretic, tuple-based semantics, [14,17,27] present results on complexity of CQA for conjunctive queries, functional dependencies and foreign key constraints. A majority semantics was studied in [28] for database merging.

The range semantics for CQA of aggregate queries was introduced and investigated in [11]. In that paper, the *NP*-completeness of CQA for atomic aggregate queries, tuple-based and set-oriented repairs, and functional dependencies was established.

Previous research reported in [3,6] is the closest to our work, because changes in attribute values are basic repair actions. However, the peculiarities of numerical values and quantitative distances between databases are not investigated.

Recent research presented in [29] investigates the complexity of repair checking and CQA wrt aggregation constraints. In this case, the constraints impose linear restrictions on summarizations. The repair semantics is based on changes of numerical attribute values, as in our case. However, the distance between instances does not consider the numerical values, but the set of changes wrt cardinality or set inclusion. Queries are atomic, without aggregation. Computational mechanisms are not considered. However, in [30] the authors present a system that uses linear programming techniques for computing a repair wrt aggregation constraints. The repair minimizes the number of changes of attribute values.

In [24], optimizations, the implementation, and experiments of/with the approximation algorithm for *DROP* are presented.

A repair semantic based on changes of attribute values is also considered in [31]. The ICs considered are functional and inclusion dependencies. Database tuples have numerical weights that may reflect provenance in data integration. In consequence, a repair has a weight that reflects the weights of the tuples modified by it. Except for these external weights, numerical attributes values are not investigated. The authors concentrate of developing and investigating heuristics for computing minimum cost repairs, but CQA is not addressed.

There is interesting work in the area of *statistical data editing* [26]. Similar to integrity constraints, *edits* are used to express conditions that a data set should satisfy [32]. Edits can be expressed as linear inequalities. There are several alternative ways of modifying the data so that edits are satisfied [25,32–34]. Those methods are tailored to finding a single ‘‘repair’’. CQA has not been considered in that area, and, to the best of our knowledge, the complexity of the problem has not been investigated.

Acknowledgments

Research supported by NSERC, 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 Lab.). Part of this work was done while L. Bravo was doing her Ph.D. at Carleton University. A. Lopatenko was working at the Free University of Bozen-Bolzano and studying at the University of the Manchester while doing this research. Extremely valuable feedback from anonymous referees is much appreciated.

References

[1] L. Bertossi, J. Chomicki, Query answering in inconsistent databases, in: J. Chomicki, G. Saake, R. van der Meyden (Eds.), Logics for Emerging Applications of Databases, Springer, Berlin, 2003.

[2] M. Arenas, L. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proceedings of the ACM Symposium on Principles of Database Systems (PODS 99), 1999, pp. 68–79.

[3] E. Franconi, A. Laureti Palma, N. Leone, S. Perri, F. Scarcello, Census data repair: a challenging application of disjunctive logic programming, in: Proceedings of Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 01), Lecture Notes in Computer Science, vol. 2250, Springer, Berlin, 2001, pp. 561–578.

[4] L. Bertossi, Consistent query answering in databases, ACM Sigmod Record 35 (2) (2006) 68–76.

[5] J. Chomicki, Consistent query answering: five easy pieces, in: Proceedings of the 11th International Conference on Database Theory (ICDT 07), Lecture Notes in Computer Science, vol. 4353, Springer, Berlin, 2007, pp. 1–17.

[6] J. Wijsen, Condensed representation of database repairs for consistent query answering, in: Proceedings of the International Conference on Database Theory (ICDT 03), Lecture Notes in Computer Science, vol. 2572, Springer, Berlin, 2003, pp. 378–393.

[7] J. Wijsen, Making more out of an inconsistent database, in: Proceedings of East-European Conference on Advances in Databases and Information Systems (ADBIS 04), Lecture Notes in Computer Science, vol. 3255, Springer, Berlin, 2004, pp. 291–305.

[8] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, Reading, MA, 1995.

[9] Ch. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.

[10] G. Kuper, L. Libkin, J. Paredaens (Eds.), Constraint Databases, Springer, Berlin, 2000.

[11] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, J. Spinrad, Scalar aggregation in inconsistent databases, Theoret. Comput. Sci. 296 (2003) 405–434.

[12] M. Krentel, The complexity of optimization problems, J. Comput. Systems Sci. 36 (1988) 490–509.

[13] D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS, 1997.

[14] J. Chomicki, J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, Inform. Comput. 197 (1–2) (2005) 90–121.

[15] C. Lund, M. Yannakakis, On the hardness of approximating minimization problems, J. Assoc. Comput. Machinery 45 (5) (1994) 960–981.

[16] V. Chvatal, A greedy heuristic for the set covering problem, Math. Oper. Res. 4 (1979) 233–235.

[17] A. Fuxman, R. Miller, First-order query rewriting for inconsistent databases, J. Comput. System Sci. 73 (4) (2007) 537–690.

[18] M. Garey, D. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (3) (1976) 237–267.

[19] M. Halldorsson, J. Radhakrishnan, Greed is good: approximating independent sets in sparse and bounded-degree graphs, in: Proceedings of the ACM Symposium on Theory of Computing (SToC 94), 1994, pp. 439–448.

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

[21] V. Vazirani, Approximation Algorithms, Springer, Berlin, 2001.

[22] L. Bertossi, L. Bravo, E. Franconi, A. Lopatenko, The complexity and approximation of fixing numerical attributes in databases under integrity constraints, Extended version with appendices ⟨http://www.scs.carleton.ca/~bertossi/papers/longIS07.pdf⟩.

[23] K. Ross, D. Srivastava, P. Stuckey, S. Sudarshan, Foundations of aggregation constraints, Theoret. Comput. Sci. 193 (1–2) (1998) 149–179.

[24] A. Lopatenko, L. Bravo, Efficient approximation algorithms for repairing inconsistent databases, in: Proceedings of the International Conference on Data Engineering (ICDE'03), 2003, pp. 216–225.

[25] A. Boskovitz, R. Goré, P. Wong, Data editing and logic, in: Proceedings of the Work Session on Statistical Data Editing, United Nations Statistical Commission and Economic Commission for Europe, Conference of European Statisticians, Ottawa, Canada, May 2005 ⟨http://www.unece.org/stats/documents/2005.05.sde.htm⟩.

[26] United Nations Statistical Commission and Economic Commission for Europe. Glossary of Terms on Statistical Data Editing. Conference of European Statisticians, Methodological Material, United Nations, Geneva, 2000, 18pp ⟨http://www.unece.org/stats/publications/editingglossary.pdf⟩.

[27] A. Cali, D. Lembo, R. Rosati, On the decidability and complexity of query answering over inconsistent and incomplete databases, in: Proceedings of the ACM Symposium on Principles of Database Systems (PODS 03), 2003, pp. 260–271.

[28] J. Lin, A.O. Mendelzon, Merging databases under constraints, Int. J. Cooperative Inform. Syst. 7 (1) (1996) 55–76.

[29] S. Flesca, F. Furfaro, F. Parisi, Consistent query answers on numerical databases under aggregate constraints, in: Proceedings of the Tenth International Symposium on Database Programming Languages (DBPL 05), Lecture Notes in Computer Science, vol. 3774, Springer, Berlin, 2005, pp. 279–294.

[30] B. Fazzinga, S. Flesca, F. Furfaro, F. Parisi, DART: a data acquisition and repairing tool, in: Current Trends in Database Technology. EDBT 2006 Workshops (Proceedings of the IIDB 06), Lecture Notes in Computer Science, vol. 4254, Springer, Berlin, 2006, pp. 297–317.

[31] P. Bohannon, W. Fan, M. Flaster, R. Rastogi, A cost-based model and effective heuristic for repairing constraints by value modification, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD 05), 2005, pp. 143–154.

[32] P. Fellegi, D. Holt, A systematic approach to automatic edit and imputation, J. Amer. Statist. Assoc. 71 (353) (1976) 17–35.

[33] R. Bruni, Error correction for massive data sets, Optim. Methods Software 20 (2–3) (2005) 295–314.

[34] A. Boskovitz, R. Goré, M. Hegland, A logical formalisation of the fellegi-holt method of data cleaning, in: Lecture Notes in Computer Science, vol. 2810, Springer, Berlin, 2003, pp. 554–565.