Complexity and Approximation of Fixing Numerical Attributes in Databases Under Integrity Constraints[⋆]

Leopoldo Bertossi[1], Loreto Bravo[1],
Enrico Franconi[2], and Andrei Lopatenko[2,⋆⋆]

[1] Carleton University, School of Computer Science, Ottawa, Canada
{bertossi, lbravo}@scs.carleton.ca
[2] Free University of Bozen–Bolzano, Faculty of Computer Science, Italy
{franconi, lopatenko}@inf.unibz.it

Abstract. Consistent query answering is the problem of computing the answers from a database that are consistent with respect to certain integrity constraints that the database as a whole may fail to satisfy. Those answers are characterized as those that are invariant under minimal forms of restoring the consistency of the database. In this context, we study the problem of repairing databases by fixing integer numerical values at the attribute level with respect to denial and aggregation constraints. We introduce a quantitative definition of database fix, and investigate the complexity of several decision and optimization problems, including *DFP*, i.e. the existence of fixes within a given distance from the original instance, and *CQA*, i.e. deciding consistency of answers to aggregate conjunctive queries under different semantics. We provide sharp complexity bounds, identify relevant tractable cases; and introduce approximation algorithms for some of those that are intractable. More specifically, we obtain results like undecidability of existence of fixes for aggregation constraints; *MAXSNP*-hardness of *DFP*, but a good approximation algorithm for a relevant special case; and intractability but good approximation for *CQA* for aggregate queries for one database atom denials (plus built-ins).

1 Introduction

Integrity constraints (ICs) are used to impose semantics on a database with the purpose of making the database an accurate model of an application domain. Database management systems or application programs enforce the satisfaction of the ICs by rejecting undesirable updates or executing additional compensating actions. However, there are many situations where we need to interact with databases that are inconsistent in the sense that they do not satisfy certain desirable ICs. In this context, an important problem in database research consists in characterizing

⋆ Dedicated to the memory of Alberto Mendelzon. Our research on this topic started with conversations between Loreto Bravo and him. Alberto was always generous with his time, advice and ideas; our community is already missing him very much.
⋆⋆ Also: University of Manchester, Department of Computer Science, UK.

and retrieving consistent data from inconsistent databases [4], in particular consistent answers to queries. From the logical point of view, consistently answering a query posed to an inconsistent database amounts to evaluating the truth of a formula against a particular *class of first-order structures* [2], as opposed to the usual process of truth evaluation in a single structure (the relational database).

Certain database applications, like census, demographic, financial, and experimental data, contain quantitative data, usually associated to nominal or qualitative data, e.g. number of children associated to a household identification code (or address); or measurements associated to a sample identification code. Usually 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 exceeds 20; or if the age of a parent is less than 10. These restrictions can be expressed with denial integrity constraints, that prevent some attributes from taking certain values [11]. Other restrictions may be expressed with aggregation ICs, e.g. the maximum concentration of certain toxin in a sample may not exceed a certain specified amount; or the number of married men and married women must be the same. 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.

We consider the problem of fixing integer numerical data wrt certain constraints while (a) keeping the values for the attributes 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 fixing processes. We concentrate on denial and aggregation 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] (c.f. [4] for a survey). There, consistent data is characterized as invariant under all minimal forms of restoring consistency, i.e. as data that is present in all minimally repaired versions of the original instance (the *repairs*). Thus, 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 minimal repair. However, in certain applications it may make more sense to correct (update) numerical values only in certain attributes. 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.

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

Example 1. Consider a network traffic database D that stores flow measurements of links in a network. This network has two types of links, labelled 0 and 1, with maximum capacities 1000 and 1500, resp. Database D is inconsistent wrt this IC. 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*$(1.1, a, 1, 1100)$; satisfying an implicit requirement that the numbers should not change too much. □

Update-based repairs for restoring consistency are studied in [25]; where changing values in attributes in a tuple is made a primitive repair action; and 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 [25, 26] is based on set-theoretic homomorphisms, but not quantitative, as in this paper. In [25] the repaired instances are called *fixes*, a term that we keep here (instead of *repairs*), because our basic repair actions are also changes of (numerical) attribute values. In this paper we consider fixable attributes that take integer values and the quadratic, Euclidean distance L_2 between database instances. Specific fixes 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. However, moving to the case of real numbers will certainly bring new issues that 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 [18].

The problem of attribute-based correction of census data forms is addressed in [11] 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 [11].

We provide semantic foundations for fixes that are based on changes on numerical attributes in the presence of key dependencies and wrt denial and aggregate ICs, 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 Fix Problem" (*DFP*), of determining the existence of a fix at a distance not bigger than a given bound, in particular considering the problems of construction and verification of such a fix. These problems are highly relevant for large inconsistent databases. For example, solving *DFP* can help us find the minimum distance from a fix to the original instance; information that can be used to prune impossible branches in the process of materialization of a fix. The *CQA* problem of deciding the consistency of query answers is studied wrt decidability, complexity, and approximation under several alternative semantics.

We prove that *DFP* and *CQA* become undecidable in the presence of aggregation constraints. However, *DFP* is *NP*-complete for linear denials, which are enough to capture census like applications. *CQA* belongs to Π_2^P and becomes Δ_2^P-hard, but for a relevant class of denials we get tractability of CQA to non aggregate queries, which is again lost with aggregate queries. Wrt approximation, we prove that *DFP* is *MAXSNP*-hard in general, and for a relevant subclass of denials we provide an approximation within a constant factor that depends on the number of atoms in them. All the algorithmic and complexity results, unless otherwise stated, refer to data complexity [1], i.e. to the size of the database that here includes a binary representation for numbers. For complexity theoretic definitions and classical results we refer to [21].

This paper is structured as follows. Section 2 introduces basic definitions. Sections 3 presents the notion of database fix, several notions of consistent answer to a query; and some relevant decision problems. Section 4 investigates their complexity. In Section 5 approximations for the problem of finding the minimum distance to a fix are studied, obtaining negative results for the general case, but good approximation for the 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 presents some conclusions and refers to related work. Proofs and other auxiliary, technical results can be found in [5].

2 Preliminaries

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

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

A *linear denial constraint* [18] has the form $\forall \bar{x} \neg (A_1 \wedge \ldots \wedge A_m)$, where the A_i are database atoms (i.e. with predicate in $\mathcal{R}$), or built-in atoms of the form $x\theta c$, where x is a variable, c is a constant and $\theta \in \{=, \neq, <, >, \leq, \geq\}$, or $x = y$. If $x \neq y$ is allowed, we call them *extended* linear denials.

Example 2. The following are linear denials (we replace $\wedge$ by a comma): (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 60000 has "silver" status: $\forall Id, Age, Income, Status \neg(Customer(Id, Age, Income, Status),$

[1] With simple denial constraints, numbers can be restricted to, e.g. $\mathbb{N}$ or $\{0, 1\}$.

Income < 60000, *Status* $=$ *silver*). (c) The constraints in Example 1, e.g. $\forall T, L$, *Type*, *Flow*$\neg$(*Traffic*$(T, L,$ *Type*, *Flow*), *Type* $= 0$, *Flow* > 1000). □

We consider aggregation constraints (ACs) [23] and aggregate queries with *sum, count, average*. *Filtering* ACs impose conditions on the tuples over which aggregation is applied, e.g. $sum(A_1 : A_2 = 3) > 5$ is a sum over A_1 of tuples with $A_2 = 3$. *Multi-attribute* ACs allow 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 relation, it is *multi-relation*, e.g. $sum_{R_1}(A_1) = sum_{R_2}(A_1)$, otherwise it is *single-relation*.

An *aggregate conjunctive query* has the form $q(x_1, \dots x_m;\ agg(z)) \leftarrow B(x_1, \dots, x_m, z, y_1, \dots, y_n)$, where *agg* is an aggregation function and its *non-aggregate matrix* (NAM) given by $q'(x_1, \dots x_m) \leftarrow B(x_1, \dots, x_m,\ z,\ y_1, \dots,\ y_n)$ is a usual first-order (FO) conjunctive query with built-in atoms, such that the *aggregation attribute* z does not appear among the x_i. Here we use the set semantics. An aggregate conjunctive query is *cyclic* (*acyclic*) if its NAM is cyclic (acyclic) [1].

Example 3. $q(x, y, sum(z)) \leftarrow R(x, y), Q(y, z, w),\ w \neq 3$ 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 $\{(1, 2, 5 + 6), (2, 3, 1 + 1)\}$.□

An *aggregate comparison query* is a sentence of the form $q(agg(z)), agg(z)\theta k$, where $q(agg(z))$ is the head of a scalar aggregate conjunctive query (with no free variables), 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 bigger than 5: $Q\colon q(sum(z)), sum(z) > 5$, with $q(sum(z)) \leftarrow R(x, y), Q(y, z, w), w \neq 3$.

3 Least Squares Fixes

When we update numerical values to restore consistency, it is desirable to make the smallest overall variation of the original values, while considering the relative relevance or specific scale of each of the fixable attributes. Since the original instance and a fix will share the same key values (c.f. Definition 2), we can use them to compute variations in the numerical values. For a tuple $\bar{k}$ of values for the key K_R of relation R in an instance D, $\bar{t}(\bar{k}, R, D)$ denotes the unique tuple $\bar{t}$ in relation R in instance D whose key value is $\bar{k}$. To each attribute $A \in \mathcal{F}$ a fixed numerical weight α_A is assigned.

Definition 1. For instances D and D' over schema Σ with the same set $val(K_R)$ of tuples of key values for each relation $R \in \mathcal{R}$, their *square distance* is

$$\Delta_{\bar{\alpha}}(D, D') = \sum_{\substack{R \in \mathcal{R}, A \in \mathcal{F} \\ \bar{k} \in val(K_R)}} \alpha_A (\pi_A(\bar{t}(\bar{k}, R, D)) - \pi_A(\bar{t}(\bar{k}, R, D')))^2,$$

where π_A is the projection on attribute A and $\bar{\alpha} = (\alpha_A)_{A \in \mathcal{F}}$. □

Definition 2. For an instance D, a set of fixable attributes $\mathcal{F}$, a set of key dependencies $\mathcal{K}$, such that $D \models \mathcal{K}$, and a set of flexible ICs IC: A *fix* for D wrt IC is an instance D' such that: (a) D' has the same schema and domain as D; (b) D' has the same values as D in the attributes in $\mathcal{A} \smallsetminus \mathcal{F}$; (c) $D' \models \mathcal{K}$; and (d) $D' \models IC$. A *least squares fix* (LS-fix) for D is a fix D' that minimizes the square distance $\Delta_{\bar{\alpha}}(D, D')$ over all the instances that satisfy (a) - (d). □

In general we are interested in LS-fixes, but (non-necessarily minimal) fixes will be useful auxiliary instances.

Example 4. (example 1 cont.) $\mathcal{R} = \{\mathit{Traffic}\}$, $\mathcal{A} = \{Time, Link, Type, Flow\}$, $K_{\mathit{Traffic}} = \{Time, Link\}$, $\mathcal{F} = \{Type, Flow\}$, with weights $\bar{\alpha} = (10^{-5}, 1)$, resp. For original instance D, $val(K_{\mathit{Traffic}}) = \{(1.1, a), (1.1, b), (1.3, b)\}$, $\bar{t}((1.1, a), \mathit{Traffic}, D) = (1.1, a, 0, 1100)$, etc. Fixes are $D_1 = \{(1.1, a, 0, 1000), (1.1, b, 1, 900), (1.3, b, 1, 850)\}$ and $D_2 = \{(1.1, a, 1, 1100), (1.1, b, 1, 900), (1.3, b, 1, 850)\}$, with distances $\Delta_{\bar{\alpha}}(D, D_1) = 100^2 \times 10^{-5} = 10^{-1}$ and $\Delta_{\bar{\alpha}}(D, D_2) = 1^2 \times 1$, resp. Therefore, D_1 is the only LS-fix. □

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 $A \in \mathcal{F}$ and $\Delta_{\bar{\alpha}}(D, D')$ will be simply denoted by $\Delta(D, D')$.

Example 5. The database D has relations $\mathit{Client}(ID, A, M)$, with key ID, attributes A for age and M for amount of money; and $\mathit{Buy}(ID, I, P)$, with key $\{ID, I\}$, I for items, and P for prices. We have denials $IC_1 : \forall ID, P, A, M \neg (\mathit{Buy}(ID, I, P), \mathit{Client}(ID, A, M), A < 18,\ P > 25)$ and $IC_2 : \forall ID, A, M \neg (\mathit{Client}(ID, A, M), A < 18,\ M > 50)$, requiring that people younger than 18 cannot spend more than 25 on one item nor spend more than 50 in the store. We added an extra column in the tables with a label for each tuple. IC_1 is violated by $\{t_1, t_4\}$ and $\{t_1, t_5\}$; and IC_2 by $\{t_1\}$ and $\{t_2\}$. We have two LS-fixes (the modified version of tuple t_1 is t_1',

D:

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

D':

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

D'':

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

etc.), with distances $\Delta(D, D') = 2^2 + 1^2 + 2^2 + 1^2 = 10$, and $\Delta(D, D'') = 3^2 + 1^2 = 10$. We can see that a global fix may not be the result of applying "local" minimal fixes to tuples. □

The built-in atoms in linear denials determine a solution space for fixes as an intersection of semi-spaces, and LS-fixes can be found at its "borders" (c.f. previous example and Proposition A.1 in [5]). It is easy to construct examples with an exponential number of fixes. For the kind of fixes and ICs we are considering, it is possible that no fix exists, in contrast to [2, 3], where, if the set of ICs is consistent as a set of logical sentences, a fix for a database always exist.

Example 6. $R(X, Y)$ has key X and fixable Y. $IC_1 = \{\forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 2), \forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 1, X_2 = 3), \forall X_1 X_2 Y \neg(R(X_1, Y), R(X_2, Y), X_1 = 2, X_2 = 3), \forall XY \neg(R(X, Y), Y > 3), \forall XY \neg(R(X, Y), Y < 2)\}$ is consistent. The first three ICs force Y to be different in every tuple. The last two ICs require $2 \leq Y \leq 3$. The inconsistent database $R = \{(1, -1), (2, 1), (3, 5)\}$ has no fix. Now, for IC_2 with $\forall X, Y \neg(R(X, Y), Y > 1)$ and $sum(Y) = 10$, any database with less than 10 tuples has no fixes. □

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

4 Decidability and Complexity

In applications where fixes are based on changes of numerical values, computing concrete fixes is a relevant problem. In databases containing census forms, correcting the latter before doing statistical processing is a common problem [11]. In databases with experimental samples, we can fix certain erroneous quantities as specified by linear ICs. In these cases, the fixes are relevant objects to compute explicitly, which contrasts with CQA [2], where the main motivation for introducing repairs is to formally characterize the notion of a consistent answer to a query as an answer that remains under all possible repairs. In consequence, we now consider some decision problems related to existence and verification of LS-fixes, and to CQA under different semantics.

Definition 3. For an instance D and a set IC of ICs:
(a) $Fix(D, IC) := \{D' \mid D'$ is an LS-fix of D wrt $IC\}$, the *fix checking problem.*
(b) $Fix(IC) := \{(D, D') \mid D' \in Fix(D, IC)\}$.
(c) $NE(IC) := \{D \mid Fix(D, IC) \neq \emptyset\}$, for *non-empty* set of fixes, i.e. the problem of *checking existence of LS-fixes.*
(d) $NE := \{(D, IC) \mid Fix(D, IC) \neq \emptyset\}$.
(e) $DFP(IC) := \{(D, k) \mid$ there is $D' \in Fix(D, IC)$ with $\Delta(D, D') \leq k\}$, the *database fix problem*, i.e. the problem of checking existence of LS-fixes within a given positive distance k.
(f) $DFOP(IC)$ is the optimization problem of finding the minimum distance from an LS-fix wrt IC to a given input instance. □

Definition 4. Let D be a database, IC a set ICs, and Q a conjunctive query[2].
(a) A ground tuple $\bar{t}$ is a *consistent answer* to $Q(\bar{x})$ under the: (a1) *skeptical semantics* if for every $D' \in Fix(D, IC)$, $D' \models Q(\bar{t})$. (a2) *brave semantics* if there

[2] Whenever we say just "conjunctive query" we understand it is a non aggregate query.

exists $D' \in \mathit{Fix}(D, \mathit{IC})$ with $D' \models Q(\bar{t})$. (a3) *majority semantics* if $|\{D' \mid D' \in \mathit{Fix}(D, \mathit{IC})$ and $D' \models Q(\bar{t})\}| > |\{D' \mid D' \in \mathit{Fix}(D, \mathit{IC})$ and $D' \not\models Q(\bar{t})\}|$.
(b) That $\bar{t}$ is a consistent answer to Q in D under semantics $\mathcal{S}$ is denoted by $D \models_{\mathcal{S}} Q[\bar{t}]$. If Q is ground and $D \models_{\mathcal{S}} Q$, we say that *yes* is a consistent answer, meaning that Q is true in the fixes of D according to semantics S. $\mathit{CA}(Q, D, \mathit{IC}, \mathcal{S})$ is the set of consistent answers to Q in D wrt IC under semantics $\mathcal{S}$. For ground Q, if $\mathit{CA}(Q, D, \mathit{IC}, \mathcal{S}) \neq \{\mathit{yes}\}$, $\mathit{CA}(Q, D, \mathit{IC}, \mathcal{S}) := \{\mathit{no}\}$.
(c) $\mathit{CQA}(Q, \mathit{IC}, \mathcal{S}) := \{(D, \bar{t}) \mid \bar{t} \in \mathit{CA}(Q, D, \mathit{IC}, \mathcal{S})\}$ is the decision *problem of consistent query answering*, of checking consistent answers. □

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. □

In Proposition 2, it suffices for queries *False*, *True* to be false, resp. true, in all instances that share the key values with the input database. Then, they can be represented by $\exists Y R(\bar{c}, Y)$, where $\bar{c}$ are not (for *False*), or are (for *True*) key values in the original instance.

Theorem 1. Under extended linear denials and complex, filtering, multi-attribute, single-relation, aggregation constraints, the problems *NE* of existence of LS-fixes, and *CQA* under skeptical or majority semantics are undecidable. □

The result about *NE* can be proved by reduction from the undecidable Hilbert's problem on solvability of diophantine equations. For CQA, apply Proposition 2. Here we have the original database and the set of ICs as input parameters. In the following we will be interested in data complexity, when only the input database varies and the set of ICs is fixed [1].

Theorem 2. For a fixed set *IC* of linear denials: (a) Deciding if for an instance D there is an instance D' (with the same key values as D) that satisfies *IC* with $\Delta(D, D') \leq k$, with positive integer k that is part of the input, is in *NP*. (b) *DFP*(*IC*) is *NP*-complete. (c.f. Definition 3(e)) □

By Proposition 1, there is a fix for D wrt *IC* at a distance $\leq k$ iff there is an LS-fix at a distance $\leq k$. Part (b) of Theorem 2 follows from part (a) and a reduction of *Vertex Cover* to $\mathit{DFP}(\mathit{IC}_0)$, for a fixed set of denials IC_0. By Theorem 2(a), if there is a fix at a distance $\leq k$, the minimum distance to D for a fix can be found by binary search in $\mathit{log}(k)$ steps. Actually, if an LS-fix exists, its square distance to D is polynomially bounded by the size of D (c.f. proof of Theorem 3 in [5]). Since D and a fix have the same number of tuples, only the size of their values in a fix matter, and they are constrained by a fixed set of linear denials and the condition of minimality.

Theorem 3. For a fixed set *IC* of extended linear denials: (a) The problem *NE*(*IC*) of deciding if an instance has an LS-fix wrt *IC* is *NP*-complete, and (b) *CQA* under the skeptical and the majority semantics is *coNP*-hard. □

For hardness in (a), (b) in Theorem 3, linear denials are good enough. Membership in (a) can be obtained for any fixed finite set of extended denials. Part (b) follows from part (a). The latter uses a reduction from *3-Colorability*.

Theorem 4. For a fixed set *IC* of extended linear denials: (a) The problem *Fix*(*IC*) of checking if an instance is an LS-fix is *coNP*-complete, and (b) *CQA* under skeptical semantics is in Π_2^P, and, for ground atomic queries, Δ_2^P-hard. □

Part (a) uses *3SAT*. Hardness in (b) is obtained by reduction from a Δ_2^P-complete decision version of the problem of searching for the lexicographically *Maximum 3-Satisfying Assignment* (*M3SA*): Decide if the last variable takes value 1 in it [17–Theo. 3.4]. Linear denials suffice. Now, by reduction from the *Vertex Cover Problem*, we obtain.

Theorem 5. For aggregate comparison queries using *sum*, *CQA* under linear denials and brave semantics is *coNP*-hard. □

5 Approximation for the Database Fix Problem

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

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

This result is obtained by establishing an *L*-reduction to *DFOP*(*IC*) from the *MAXSNP*-complete [22, 21] *B-Minimum Vertex Cover Problem*, i.e. the vertex cover minimization problem for graphs of bounded degree [16–Chapter 10]. As an immediate consequence, we obtain that *DFOP*(*IC*) cannot be uniformly approximated within arbitrarily small constant factors [21].

Corollary 1. Unless $P = NP$, there is no *Polynomial Time Approximation Schema* for *DFOP*. □

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

5.1 Local Denials

Definition 5. A set of linear denials *IC* is *local* if: (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 attribute A appears in *IC* both in comparisons of the form $A < c_1$ and $A > c_2$.[3] □

In Example 5, *IC* is local. In Example 6, IC_1 is not local. Local constraints have the property that by doing local fixes, no new inconsistencies are generated, and there is always an LS-fix wrt to them (c.f. Proposition A.2 in [5]). Locality is a sufficient, but not necessary condition for existence of LS-fixes as can be seen from the database $\{P(a, 2)\}$, with the first attribute as a key and non-local denials $\neg(P(x, y), y < 3), \neg(P(x, y), y > 5)$, that has the LS-fix $\{P(a, 3)\}$.

[3] To check condition (c), $x \leq c,\ x \geq c,\ x \neq c$ have to be expressed using $<, >$, e.g. $x \leq c$ by $x < c + 1$.

Proposition 4. For the class of local denials, *DFP* is *NP*-complete, and *DFOP* is *MAXSNP*-hard. □

This proposition tells us that the problem of finding good approximations in the case of local denials is still relevant.

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

A violation set I for ic is a minimal set of tuples that simultaneously participate in the violation of ic.

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

$S(t,t')$ contains the violation sets that include t and are solved by replacing t' for t. A local fix t' solves some of the violations due to t and minimizes the distance to t.

5.2 Database Fix Problem as a Set Cover Problem

For a fixed set IC of local denials, we can solve an instance of *DFOP* by transforming it into an instance of the *Minimum Weighted Set Cover Optimization Problem* (*MWSCP*). This problem is *MAXSNP*-hard [20, 21], and its general approximation algorithms are within a logarithmic factor [20, 9]. By concentrating on local denials, we will be able to generate a version of the *MWSCP* that can be approximated within a constant factor (c.f. Section 5.3).

Definition 8. For a database D and a set IC of local denials, $\mathcal{G}(D, IC) = (T, H)$ denotes the *conflict hypergraph* for D wrt IC [8], which has in the set T of vertices the database tuples, and in the set H of hyperedges, the violation sets for elements $ic \in IC$. □

Hyperedges in H can be labelled with the corresponding ic, so that different hyperedges may contain the same tuples. Now we build an instance of *MWSCP*.

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 set collection, and w is the weight function, is given by: (a) $U := H$, the set of hyperedges of $\mathcal{G}(D, IC)$. (b) $\mathcal{S}$ contains the $S(t,t')$, where t' is a local fix for a tuple $t \in D$. (c) $w(S(t,t')) := \Delta(\{t\},\{t'\})$. □

It can be proved that the $S(t,t')$ in this construction are non empty, and that $\mathcal{S}$ covers U (c.f. Proposition A.2 in [5]).

If for the instance $(U, \mathcal{S}, w)$ of *MWSCP* we find a minimum weight cover $\mathcal{C}$, we could think of constructing a fix by replacing each inconsistent tuple $t \in D$ by a local fix t' with $S(t,t') \in \mathcal{C}$. The problem is that there might be more than one t' and the key dependencies would not be respected. Fortunately, this problem can be circumvented.

Definition 10. Let $\mathcal{C}$ be a cover for instance $(U,\mathcal{S},w)$ of the *MWSCP* associated to D, IC. (a) $\mathcal{C}^\star$ is obtained from $\mathcal{C}$ as follows: For each tuple t with local fixes $t_1,\ldots,t_n$, $n>1$, such that $S(t,t_i) \in \mathcal{C}$, replace in $\mathcal{C}$ all the $S(t,t_i)$ by a single $S(t,t^\star)$, where $t^\star$ is such that $S(t,t^\star) = \bigcup_{i=1}^n S(t,t_i)$. (b) $D(\mathcal{C})$ is the database instance obtained from D by replacing t by t' if $S(t,t') \in \mathcal{C}^\star$. □

It holds (c.f. Proposition A.3 in [5]) that such an $S(t,t^\star) \in \mathcal{S}$ exists in part (a) of Definition 10. Notice that there, tuple t could have other $S(t,t')$ outside $\mathcal{C}$. Now we can show that the reduction to *MWSCP* keeps the value of the objective function.

Proposition 5. If $\mathcal{C}$ is an optimal cover for instance $(U,\mathcal{S},w)$ of the *MWSCP* associated to D, IC, then $D(\mathcal{C})$ is an LS-fix of D wrt IC, and $\Delta(D,D(\mathcal{C})) = w(\mathcal{C}) = w(\mathcal{C}^*)$. □

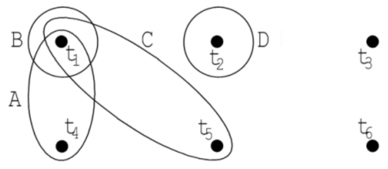
Proposition 6. For every LS-fix D' of D wrt a set of local denials IC, there exists an optimal cover $\mathcal{C}$ for the associated instance $(U,\mathcal{S},w)$ of the *MWSCP*, such that $D' = D(\mathcal{C})$. □

Proposition 7. The transformation of *DFOP* into *MWSCP*, and the construction of database instance $D(\mathcal{C})$ from a cover $\mathcal{C}$ for $(U,\mathcal{S},w)$ can be done in polynomial time in the size of D. □

We have established that the transformation of *DFOP* into *MWSCP* is an *L*-reduction [21]. Proposition 7 proves, in particular, that the number of violation sets $S(t,t')$ is polynomially bounded by the size of the original database D.

Example 7. (example 5 continued) We illustrate the reduction from *DFOP* to *MWSCP*. The violation sets are $\{t_1,t_4\}$ and $\{t_1,t_5\}$ for IC_1 and $\{t_1\}$ and $\{t_2\}$ for IC_2. The figure shows the hypergraph. For the *MWSCP* instance, we need the local fixes. Tuple t_1 has two local fixes $t_1' = (1,15,50)$, that solves the violation set $\{t_1\}$ of IC_2 (hyperedge B), and $t_1'' = (1,18,52)$, that solves the violation sets $\{t_1,t_4\}$ and $\{t_1,t_5\}$ of IC_1, and $\{t_1\}$ of IC_2 (hyperedges A,B, C), with weights 4 and 9, resp. t_2, t_4 and t_5 have one local fix each corresponding to: $(2,16,50)$, $(1, CD, 25)$ and $(1, DVD, 25)$, resp. The consistent tuple t_3 has no local fix.

The *MWSCP* instance is shown in the table, where the elements are rows and the sets (e.g. $S_1 = S(t_1,t_1')$), columns. An entry 1 means that the set contains



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

the corresponding element; and a 0, otherwise. There are two minimal covers, both with weight 10: $\mathcal{C}_1 = \{S_2, S_3\}$ and $\mathcal{C}_2 = \{S_1, S_3, S_4, S_5\}$. $D(\mathcal{C}_1)$ and $D(\mathcal{C}_2)$ are the two fixes for this problem. □

If we apply the transformation to Example 6, that had non-local set of ICs and no repairs, we will find that instance $D(\mathcal{C})$, for $\mathcal{C}$ a set cover, can be constructed as above, but it does not satisfy the flexible ICs, because changing inconsistent tuples by their local fixes solves only the initial inconsistencies, but new inconsistencies are introduced.

5.3 Approximation Via Set Cover Optimization

Now that we have transformed the database fix problem into a weighted set cover problem, we can apply approximation algorithms for the latter. We know, for example, that using a greedy algorithm, *MWSCP* can be approximated within a factor $log(N)$, where N is the size of the underlying set U [9]. The approximation algorithm returns not only an approximation $\hat{w}$ to the optimal weight w^o, but also a -non necessarily optimal- cover $\hat{\mathcal{C}}$ for problem $(U, \mathcal{S}, w)$. As in Definition 10, $\hat{\mathcal{C}}$ can be used to generate via $(\hat{\mathcal{C}})^\star$, a fix $D(\hat{\mathcal{C}})$ for D that may not be LS-minimal.

Example 8. (examples 5 and 7 continued) We show how to to compute a solution to this particular instance of *DFOP* using the greedy approximation algorithm for *MWSCP* presented in [9]. We start with $\hat{\mathcal{C}} := \emptyset$, $S_i^0 := S_i$; and we add to $\mathcal{C}$ the S_i such that S_i^0 has the maximum *contribution ratio* $|S_i^0|/w(S_i^0)$. The alternatives are $|S_1|/w(S_1) = 1/4$, $|S_2|/w(S_2) = 3/9$, $|S_3|/w(S_3) = 1$, $|S_4|/w(S_4) = 1/4$ and $|S_5|/w(S_5) = 1$. The ratio is maximum for S_3 and S_5, so we can add any of them to $\hat{\mathcal{C}}$. If we choose the first, we get $\hat{\mathcal{C}} = \{S_3\}$. Now we compute the $S_i^1 := S_i^0 \smallsetminus S_3^0$, and choose again an S_i for $\hat{\mathcal{C}}$ such that S_i^1 maximizes the contribution ratio. Now S_5 is added to $\hat{\mathcal{C}}$, because S_5^1 gives the maximum. By repeating this process until we get all the elements of U covered, i.e. all the S_i^k become empty at some iteration point k, we finally obtain $\hat{\mathcal{C}} = \{S_3, S_5, S_1, S_4\}$. In this case $\hat{\mathcal{C}}$ is an optimal cover and therefore, $D(\hat{\mathcal{C}})$ is exactly an LS-fix, namely D' in Example 5. Since this is an approximation algorithm, in other examples the cover obtained might not be optimal. □

Proposition 8. Given database instance D with local ICs IC, the database instance $D(\hat{\mathcal{C}})$ obtained from the approximate cover $\hat{\mathcal{C}}$ is a fix and it holds $\Delta(D, D(\hat{\mathcal{C}})) \leq log(N) \times \Delta(D, D')$, where D' is any LS-fix of D wrt IC and N is the number of of violation sets for D wrt IC. □

In consequence, for any set IC of local denials, we have a polynomial time approximation algorithm that solves $DFOP(IC)$ within an $O(log(N))$ factor, where N is the number of violation sets for D wrt IC. As mentioned before, this number N, the number of hyperedges in $\mathcal{G}$, is polynomially bounded by $|D|$ (c.f. Proposition 7). 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, in our case we can get even better approximations via a cover $\hat{\mathcal{C}}$ obtained with an approximation algorithms for the special case of the *MWSCP* where the number of occurrences of an element of U in elements of $\mathcal{S}$ is bounded by a constant. For this case of the *MWSCP* there are approximations within a constant factor based on "linear relaxation" [16–Chapter 3]. This is clearly the case in our application, being $m \times |\mathcal{F}| \times |IC|$ a constant bound (independent from $|D|$) on the frequency of the elements, where m is the maximum number of database atoms in an IC.

Theorem 6. There is an approximation algorithm that, for a given database instance D with local ICs IC, returns a fix $D(\hat{\mathcal{C}})$ such that $\Delta(D, D(\hat{\mathcal{C}})) \leq c \times \Delta(D, D')$, where c is a constant and D' is any LS-fix of D. $\square$

6 One Atoms 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 $\mathcal{R}$, and B is a conjunction of built-in atoms. They capture range constraints; and census data is usually stored in single relation schemas [11].

For 1ADs, we can identify tractable cases for *CQA* under LS-fixes by reduction to *CQA* for (tuple and set-theoretic) repairs of the form introduced in [2] for key constraints. This is because each violation set (c.f. Definition 6) contains one tuple, maybe with several local fixes, but all sharing the same key values; and then the problem consists in choosing one from different tuples with the same key values (c.f. proof in [5] of Theorem 7). The transformation preserves consistent answers to both ground and open queries.

The "classical" -tuple and set oriented- repair problem as introduced in [2] has been studied in detail for functional dependencies in [8, 12]. In particular, for tractability of *CQA* in our setting, we can use results and algorithms obtained in [12] for the classical framework.

The *join graph* $\mathcal{G}(Q)$ [12] of a conjunctive query Q is a directed graph, whose vertices are the database atoms in Q. There is an arc from L to L' if $L \neq L'$ and there is a variable w that occurs at the position of a non-key attribute in L and also occurs in L'. Furthermore, there is a self-loop at L if there is a variable that occurs at the position of a non-key attribute in L, and at least twice in L.

When Q does not have repeated relations symbols, we write $Q \in \mathcal{C}_{Tree}$ if $\mathcal{G}(Q)$ is a forest and every non-key to key join of Q is full i.e. involves the whole key. Classical *CQA* is tractable for queries in $\mathcal{C}_{Tree}$ [12].

Theorem 7. For a fixed set of 1ADs and queries in C_{Tree}, consistent query answering under LS-fixes is in *PTIME*. $\square$

We may define that a aggregate conjunctive query belongs to C_{Tree} if its underlying non-aggregate conjunctive query, i.e. its NAM (c.f. Section 2) belongs to C_{Tree}. Even for 1ADs, with simple comparison aggregate queries with *sum*, tractability is lost under the brave semantics.

Proposition 9. For a fixed set of 1ADs, and for aggregate queries that are in C_{Tree} or acyclic, *CQA* is *NP*-hard under the brave semantics. $\square$

For queries Q returning numerical values, which is common in our framework, it is natural to use the *range semantics* for *CQA*, introduced in [3] for scalar aggregate queries and functional dependencies under classical repairs. Under this semantics, a consistent answer is the pair consisting of the *min-max* and *max-min* answers, i.e. the supremum and the infimum, resp., of the set of answers to Q obtained from LS-fixes. The *CQA* decision problems under range semantics consist in determining if a numerical query Q, e.g. an aggregate query, has its answer $\leq k_1$ in every fix (*min-max* case), or $\geq k_2$ in every fix (*max-min* case).

Theorem 8. For each of the aggregate functions *sum, count distinct*, and *average*, there is a fixed set of 1ADs and a fixed aggregate acyclic conjunctive query, such that *CQA* under the range semantics is *NP*-hard. $\square$

For the three aggregate functions one 1AD suffices. The results for *count distinct* and *average* are obtained by reduction from *MAXSAT* [21] and *3SAT*, resp. For *sum*, we use a reduction from the *Independent Set Problem* with bounded degree 3 [14]. The general *Independent Set Problem* has bad approximation properties [16–Chapter 10]. The *Bounded Degree Independent Set* has efficient approximations within a constant factor that depends on the degree [15].

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 a constant factor for *CQA* under *min-max* range semantics. $\square$

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 [13, 24]), for which a polynomial time approximation similar to the one for *MAXSAT* can be given [24].

7 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 consistent query answering. These problems are particularly relevant in census like applications, where the problem of *data editing* is a common and difficult task (c.f. `http://www.unece.org/stats/documents/2005.05.sde.htm`). 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 extensions are in development. 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 is a subject of ongoing research.

The framework established in this paper could be applied to qualitative attributes with an implicit linear order given by the application. 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 fixes. We have developed (but not reported here) extensions to our approach that consider *minimum distribution variation* LS-fixes that keep the overall statistical properties of the database. We have also developed optimizations of the approximation algorithm presented in Section 5; and its implementation and experiments are ongoing efforts. More research on the impact of aggregation constraints on LS-fixes is needed.

Of course, if instead of the L_2 distance, the L_1 distance is used, we may get for the same database a different set of (now L_1) fixes. The actual approximations obtained in this paper change too. However, the general complexity and approximability results should remain. They basically depend on the fact that distance functions are non-negative, additive wrt attributes and tuples, computable in polynomial time, and monotonically increasing. Another possible semantics could consider an epsilon of error in the distance in such a way that if, for example, the distance of a fix is 5 and the distance to another fix is 5.001, we could take both of them as (minimal) LS-fixes.

Other open problems refer to cases of polynomial complexity for linear denials with more that one database atom; approximation algorithms for the *DFOP* for non-local cases; and approximations to CQA for other aggregate queries.

For related work, we refer to the literature on consistent query answering (c.f. [4] for a survey and references). Papers [25] and [11] are the closest to our work, because changes in attribute values are basic repair actions, but the peculiarities of numerical values and quantitative distances between databases are not investigated. Under the set-theoretic, tuple-based semantics, [8, 7, 12] report on complexity issues for conjunctive queries, functional dependencies and foreign key constraints. A majority semantics was studied in [19] for database merging. Quite recent papers, but under semantics different than ours, report research on fixing numerical values under aggregation constraints [10]; and heuristic construction of repairs based on attribute values changes [6].

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.).

References

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

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

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

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

[5] Bertossi, L., Bravo, L., Franconi, E. and Lopatenko, A. Fixing Numerical Attributes Under Integrity Constraints. Corr Archiv paper cs.DB/0503032; March 15, 2005.

[6] Bohannon, P., Michael, F., Fan, F. and Rastogi, R. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *Proc. ACM International Conference on Management of Data (SIGMOD 05)*, 2005, pp. 143-154.

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

[8] Chomicki, J. and Marcinkowski, J. Minimal-Change Integrity Maintenance Using Tuple Deletions. *Information and Computation*, 2005, 197(1-2):90-121.

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

[10] Flesca, S., Furfaro, F. and Parisi, F. Consistent Query Answers on Numerical Databases under Aggregate Constraints. In *Proc. Tenth International Symposium on Database Programming Languages (DBPL 05)*, 2005.

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

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

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

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

[15] Halldorsson, M. and Radhakrishnan, J. Greed is Good: Approximating Independent Sets in Sparse and Bounded-degree Graphs. In *Proc. ACM Symposium on Theory of Computing (SToC 94)*, 1994, pp. 439-448.

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

[17] Krentel, M. The Complexity of Optimization Problems. *J. Computer and Systems Sciences*, 1988, 36:490-509.

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

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

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

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

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

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

[24] Vazirani, V. Approximation Algorithms. Springer, 2001.

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

[26] Wijsen, J. Making More Out of an Inconsistent Database. In *Proc. East-European Conference on Advances in Databases and Information Systems (ADBIS 04)*, Springer LNCS 3255, 2004, pp. 291-305.