Generalized Weighted Repairs

Horacio Tellez Perez and Jef Wijsen[0000−0001−8216−273X]

University of Mons, Belgium
{horacio.tellezperez, jef.wijsen}@umons.ac.be

Abstract. This paper deals with the problem of repairing inconsistent relational database instances in which facts are associated with non-negative weights, representing their quality or trustfulness. Given a numeric aggregation function $\mathcal{G}$, weighted repairs (or $\mathcal{G}$-repairs) are defined as inclusion-maximal consistent subinstances with maximum aggregated value. The weighted repair notion extends existing repair notions, like subset- and cardinality-repairs. We study the complexity of repair-checking and some related problems, in a setting where database integrity constraints are represented by a hypergraph whose hyperedges correspond to constraint violations. In this setting, $\mathcal{G}$-repairs can be viewed as maximum-weight independent sets relative to the aggregation function $\mathcal{G}$.

Keywords: Conflict hypergraph · Consistent query answering · Database repairing · Maximum-weight independent set.

1 Motivation

In today's era of "big data," database management systems have to cope more and more with dirty information that is inconsistent with respect to some integrity constraints. Such integrity constraints are commonly expressed in decidable fragments of some logic, for example, as dependencies [1] or ontologies in some Description Logic [4]. The term *repair* is commonly used to refer to a consistent database that is obtained from the inconsistent database by some minimal modifications. This notion was introduced twenty years ago in a seminal paper by Arenas et al. [3], and has been an active area of research ever since. In particular, the field of *Consistent Query Answering (CQA)* studies the question of how to answer database queries if multiple repairs are possible. Two surveys of this research are [6, 18].

This paper's aim is to contribute to the research in *preferred repair semantics*, whose goal is to capture more of the meaning of the data into the repairing process. To this end, we introduce and study *weighted repairs*. We will assume that database tuples are associated with numerical weights such that tuples with higher weights are preferred over tuples with lower weights. Then, among all possible repairs, weighted repairs are those with a maximum aggregated value, according to some aggregation function. We will study the relationship between the complexity of computing weighted repairs and certain properties of the aggregation function used.

The remainder of this section is an informal guided tour that introduces and motivates our research questions by means of a simple example. We start with a graph-theoretical view on database repairing.

A Graph-Theoretical Perspective on Database Repairing Consider the following relational table *TEACHES*, in which a fact $TEACHES(p, c, s)$ means that professor p teaches the course c during semester s.

TEACHES	*Prof*	*Course*	*Sem*	
	Jeff	A	fall	(f_1)
	Jeff	B	fall	(f_2)
	Ed	C	spring	(s_1)
	Rob	C	spring	(s_2)
	Rob	D	spring	(s_3)

The integrity constraints are as follows: no professor teaches more than one course in a given semester, and no course is taught by more than one professor. In terms of functional dependencies, we have: $\{Prof, Sem\} \rightarrow \{Course\}$ and $\{Course\} \rightarrow \{Prof\}$. The relation *TEACHES* violates these integrity constraints; its conflict graph is shown in Fig. 1. The vertices of the conflict graph are the facts in the relation *TEACHES*; two vertices are adjacent if together they violate some functional dependency.

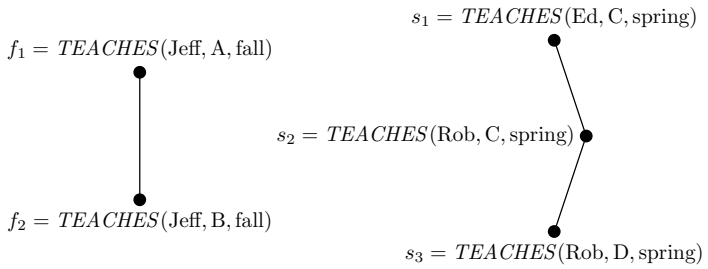

Fig. 1. Conflict graph for the running example.

Given a database instance, it is common to define a *subset-repair* as an inclusion-maximal subinstance that satisfies all integrity constraints. In terms of the conflict graph, every subset-repair is an *inclusion-maximal independent set (IMIS)*, and vice versa. Recall that in graph theory, a set of vertices is *independent* if no two vertices of it are adjacent. It can be verified that the graph of Fig. 1 has four IMISs: every IMIS includes either $\{f_1\}$ or $\{f_2\}$, and includes either $\{s_1, s_3\}$ or $\{s_2\}$. The term *cardinality-repair* refers to independent sets of maximum cardinality. In our running example, the cardinality-repairs are $\{f_1, s_1, s_3\}$ and $\{f_2, s_1, s_3\}$.

Maximum-Weight Independent Set (MWIS) As in [16], we will assume from here on that every fact is associated with a nonnegative weight, where larger weights are better. In practice, such weights may occur in data integration, where facts coming from more authoritative data sources are tagged with higher weights. For example, in the next relational table, among the first two facts—which are conflicting—the second fact has a higher weight and is therefore considered better.

TEACHES	*Prof*	*Course*	*Sem*	*w*
	Jeff	A	fall	1
	Jeff	B	fall	2
	Ed	C	spring	1
	Rob	C	spring	2
	Rob	D	spring	1

It is now natural to take these weights into account, and define repairs as maximum-weight independent sets (MWIS) of the conflict graph. In graph theory, an MWIS is an independent set that maximizes the sum of the weights of its vertices. In our example, there are two MWISs, both having a total summed weight of 4:

T_1	*Prof*	*Course*	*Sem*	*w*
	Jeff	B	fall	2
	Rob	C	spring	2

and

T_2	*Prof*	*Course*	*Sem*	*w*
	Jeff	B	fall	2
	Ed	C	spring	1
	Rob	D	spring	1

.

Aggregation Functions Other than SUM The aggregation function SUM is cooked into the definition of MWIS: among all independent sets, an MWIS is one that maximizes the *summed* weight. From a conceptual perspective, it may be natural to use aggregation functions other than SUM. For example, among the two repairs T_1 and T_2 shown above, we may prefer T_1 because it maximizes the *average* weight. Indeed, the average weights for T_1 and T_2 are, respectively, $\frac{2+2}{2}$ and $\frac{2+1+1}{3}$. Alternatively, we may prefer T_1 because it maximizes the *minimum* weight. Therefore, to capture these alternatives, we will allow other functions than SUM in this paper, including AVG and MIN.

Computing Repairs In data cleaning and database repairing, we are often interested in finding one or more repairs for a given database instance. Now that we have introduced weights and different aggregation functions, this boils down to the following task: given a database instance with weighted facts, return an inclusion-maximal consistent subinstance that maximizes the aggregated weight according to some fixed aggregation function. Alternatively, in graph-theoretical terms: given a vertex-weighted graph, return an inclusion-maximal independent set that maximizes the aggregated weight according to some fixed aggregation function. Since for aggregation functions other than SUM, maximality with respect to set inclusion and maximality with respect to aggregated weight may not go hand in hand, it should be made precise which criterion prevails:

- among all inclusion-maximal independent sets, return one that maximizes the aggregated weight; or
- among all independent sets that maximize the aggregated weight, return one that is inclusion-maximal.

To illustrate the difference, let $G = (V, E)$ with $V = \{a, b\}$ and $E = \emptyset$. Let $w(a) = 1$ and $w(b) = 2$, and let MIN be the aggregation function. The first task would return $\{a, b\}$, while the second task would return $\{b\}$. In this paper, we will study the latter task.

It is known that under standard complexity assumptions (in particular, $\mathbf{P} \neq \mathbf{NP}$), there is no polynomial-time algorithm that returns an MWIS for a given vertex-weighted graph. Therefore, when the aggregation function SUM is used, it is intractable to return a repair with a maximum summed weight. In this paper, we will ask whether this task can become tractable for other aggregation functions of practical interest. Contributions of this paper can be summarized as follows.

- We introduce $\mathcal{G}$-repairs, generalizing existing repair notions.
- By taking a conflict hypergraph perspective on database integrity, we show that $\mathcal{G}$-repairs are a generalization of maximum-weight independent sets.
- We adapt classical decision problems to our setting, and study their computational complexity. While these problems are intractable in general, we identify classes of aggregation functions that allow for polynomial-time algorithms.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces aggregation functions and defines the (conflict) hypergraph perspective for studying inconsistent databases. Section 4 defines the notion of $\mathcal{G}$-repair and the computational problems we are interested in. Section 5 shows computational complexity results for these problems, culminating in our main tractability theorem, Theorem 3. Section 6 shows that tractability is lost under a slight relaxation of the hypotheses of that theorem. Finally, Section 7 concludes the paper.

2 Related Work

In their seminal paper [3], Arenas et al. define repairs of an inconsistent database as those consistent databases that can be obtained by inserting and deleting minimal (with respect to set inclusion) sets of tuples. Since then, many variants of this earliest repair notion have been introduced, several of which are discussed in [5, 9, 18]. For any fixed repair notion, *repair checking* is the following problem: given an inconsistent database and a candidate repair, is the candidate a true repair (i.e., does it satisfy all constraints imposed by the repair notion under consideration)? Afrati and Kolaitis [2] made important contributions to our understanding of the complexity of repair checking. For databases containing numerical attributes, repairs have also been defined as solutions to numerical constraint problems [7, 8, 13].

The notion of *conflict hypergraph* was introduced in [10], and later extended in [17]. The relationship between repairs and inclusion-maximal independent sets was observed in [10, Proposition 3.1]. If database tuples are associated with nonnegative weights, then it is natural to generalize this relationship by viewing repairs as maximum-weight independent sets (MWIS). We cannot cite here the vast amount of literature studying the computational complexity and algorithms related to MWIS. In our approach, however, we do not primarily focus on the maximum *summed* weight, but also allow for aggregation functions other than SUM. The problems we study are specifically motivated by database applications in which several other aggregation functions are sensible. We will show that some problems that are **NP**-hard in general, become tractable for aggregation functions that have some desirable properties. Inspired by database theory, weight-based approaches to inconsistency have also been studied for knowledge bases in Description Logics [12].

3 Preliminaries

Aggregation Functions over Weighted Sets Informally, aggregation functions take as input a set of elements, each associated with a weight, and return an aggregated weight for the entire set. Examples are SUM and MIN. In this work, all weights will be nonnegative rational numbers, which we interpret as quality scores where higher values are better. These notions are formalized next.

Definition 1 (Weighted set). *A* weighted set *is a pair (I, w) where I is a finite set and w is a total mapping from I to $\mathbb{Q}^+$ (i.e., the set of nonnegative rational numbers). We will often assume that the* weight function *w is implicitly understood. That is, whenever we say that I is a weighted set, we mean that (I, w) is a weighted set for a mapping w that is implicitly understood.*

Two weighted sets (I_1, w_1) and (I_2, w_2) are isomorphic *if there is a bijection $\pi : I_1 \to I_2$ such that for every $x \in I_1$, $w_1(x) = w_2(\pi(x))$. Informally, two weighted sets are isomorphic if the attained numeric values as well as their multiplicities coincide.*

Definition 2 (Aggregation function). *An* aggregation function *$\mathcal{G}$ is a function that maps every weighted set (I, w) to a nonnegative rational number, denoted $\mathcal{G}_{[w]}(I)$, such that:*

- *$\mathcal{G}$ is closed under isomorphism, meaning that any two weighted sets that are isomorphic are mapped to the same value; and*
- *the empty weighted set is mapped to 0.*

We write $\mathbf{AGG}^{\mathsf{poly}}$ for the class of aggregation functions that are computable in polynomial time in $|I|$. Some well-known members of this class are denoted COUNT, SUM, MAX, MIN, *and* AVG, *with their expected, natural semantics (not repeated here).*

By measuring the complexity of an aggregation function $\mathcal{G}$ in terms of $|I|$, we do not take into account the size of the numeric values in the image of the mapping w. This complexity is appropriate for the applications we have in mind. The requirement that an aggregation function be closed under isomorphism is tantamount to saying that for a weighted set I, the value $\mathcal{G}_{[w]}(I)$ depends on, and only on, the multiset $\{\!\{w(x) \mid x \in I\}\!\}$. While it may be more common to define aggregation functions on multisets of numbers (see, e.g.,[15, p. 159]), our Definition 2 is appropriate for the purposes we have in mind. Indeed, we will only apply aggregation on weighted sets formed by vertices of a vertex-weighted graph.

Conflict Hypergraphs Conflict hypergraphs [10, 11] generalize the conflict graphs introduced previously in Section 1. To detect violations of functional dependencies, it suffices to regard two tuples at a time. However, more involved constraints may consider three or more tuples at a time. For this reason, conflict graphs are generalized to conflict hypergraphs. Informally, a conflict hypergraph is a hypergraph whose vertices are the database facts; hyperedges are formed by grouping facts that together violate some integrity constraint.

Formally, a fact is an expression $R(c_1, \ldots, c_n)$ where R is a relation name of arity n, and each c_i is a constant. A database is a finite set of facts. Let **db** be a database instance, and $\mathcal{C}$ be a set of integrity constraints. The *(conflict) hypergraph* is defined as an hypergraph $H = (V, E)$ whose vertices are the facts of **db**; there is an hyperedge $e = \{R_1(\vec{c}_1), \ldots, R_k(\vec{c}_k)\}$ if (and only if) the following properties hold:

1. the facts $R_1(\vec{c}_1), \ldots, R_k(\vec{c}_k)$ taken together violate one or more integrity constraints of $\mathcal{C}$; and
2. every strict subset of e satisfies $\mathcal{C}$.

In other words, the hyperedges of H are the inclusion-minimal inconsistent subsets of **db**. Recall from graph theory that an *independent set* of a hypergraph $H = (V, E)$ is a set $I \subseteq V$ such that I includes no hyperedge of E. Then, by construction, the following expressions are obviously equivalent for every database instance **db** and set $\mathcal{C}$ of integrity constraints:

- I is an independent set of the (conflict) hypergraph; and
- I is consistent, i.e., I satisfies $\mathcal{C}$.

It is this equivalence between independent sets and database consistency that motivates the hypergraph perspective on database repairing. For most database integrity constraints, minimal (w.r.t. $\subseteq$) inconsistent sets are bounded in size. For example, for functional dependencies or primary keys, this bound is 2. This will be mimicked in the hypergraph perspective by assuming a bound b (some positive integer) on the size of the hyperedges.

Finally, we will consider vertex-weighted hypergraphs, i.e., the vertex set will be a weighted set, as defined by Definition 1.

Definition 3. *A hypergraph is called* weighted *if its vertex set is a weighted set. Technically, such a hypergraph is a nested pair* $((V, w), E)$ *with* (V, w) *a weighted set of vertices, and* E *a set of hyperedges. However, as announced in Definition 1, we often omit the explicit mention of the weight function* w*. For simplicity, we will assume that no hyperedge is a singleton. For every integer* $b \geq 2$*, we define* $\mathbf{WH}[b]$ *as the set of weighted hypergraphs containing no hyperedge of cardinality strictly greater than* b*.*

To conclude this section, we argue that for most common database integrity constraints, the hypergraph perspective is appropriate for our computational complexity studies, even if constraints are given as expressions in relational calculus. The reason is that **P** (i.e., polynomial time) is the smallest complexity class considered in our complexity analysis, while for most database constraints, conflict hypergraphs can be obtained by a query in relational calculus, which is strictly contained in **P**. For example, for a functional dependency $R : X \to Y$, the edges of the conflict hypergraph are all pairs of tuples in R that agree on all attributes of X but disagree on some attribute in Y.

4 Repair Checking and Related Problems

A repair of an inconsistent databases **db** is often defined as a maximal consistent subinstance of **db**, where maximality can be with respect to set inclusion or cardinality, yielding subset- and cardinality-repairs, respectively. These notions carry over to the hypergraph perspective defined in Section 3. For any aggregation function $\mathcal{G}$ and weighted hypergraph, we now define $\mathcal{G}$-repairs as a natural generalization of existing repair notions. Significantly, from a graph-theoretical viewpoint, $\mathcal{G}$-repairs generalize *maximum-weight independent sets*, which are independent sets of vertices whose weights sum to the maximum possible value. In $\mathcal{G}$-repairs, other functions than SUM can be used.

Definition 4 ($\mathcal{G}$-repair). *Let* $\mathcal{G}$ *be an aggregation function, and* $H = ((V, w), E)$ *a weighted hypergraph. A* $\mathcal{G}$*-repair of* H *is a subset* $I \subseteq V$ *with the following three properties:*

Independence: *I is an independent set of H;*
Maximality: *for every other independent set $J \subseteq V$, it holds that $\mathcal{G}_{[w]}(I) \geq \mathcal{G}_{[w]}(J)$; and*
Saturation: *for every other independent set $J \subseteq V$, if $\mathcal{G}_{[w]}(I) = \mathcal{G}_{[w]}(J)$ and $I \subseteq J$, then $I = J$.*

Informally, among all independent sets that maximize $\mathcal{G}_{[w]}$, a weighted repair is one that is inclusion-maximal. Subset-repairs and cardinality-repairs are special cases of $\mathcal{G}$-repairs. Indeed, if we let $\mathcal{G} = \mathsf{SUM}$ and $w(v) = 1$ for every vertex v, then $\mathcal{G}$-repairs coincide with cardinality-repairs. If we let $\mathcal{G} = \mathsf{MIN}$ and $w(v) = 1$ for every vertex v, then $\mathcal{G}$-repairs coincide with subset-repairs.

We now relax $\mathcal{G}$-repairs by replacing the *Maximality* requirement in Definition 4 by a lower bound on the aggregated value. This corresponds to real-life situations where we may already be happy with a guaranteed lower bound.

Definition 5 (q-suitable vertex set). *This definition is relative to some fixed aggregation function $\mathcal{G}$. Let $H = ((V, w), E)$ be a weighted hypergraph, and $q \in \mathbb{Q}^+$. A set $I \subseteq V$ is said to be a* q-suitable set *of H if the following three properties hold true:*

Independence: *I is an independent set of H;*
Suitability: *$\mathcal{G}_{[w]}(I) \geq q$; and*
Saturation: *for every other independent set $J \subseteq V$ such that $I \subseteq J$, if $\mathcal{G}_{[w]}(I) \leq \mathcal{G}_{[w]}(J)$, then $I = J$.*

Informally, an independent set I is q-suitable if its aggregated value is at least q and every strict extension of I is either not independent or has a strictly smaller aggregated value. The decision problems of our interest generalize repair checking, which is central in consistent query answering [18].

Definition 6. *The following problems are relative to an aggregation function $\mathcal{G}$ in $\mathbf{AGG}^{\mathsf{poly}}$ and a positive integer b.*

PROBLEM REPAIR-CHECKING$(\mathcal{G}, b)$
Input: *A weighted hypergraph H in $\mathbf{WH}[b]$; a set I of vertices.*
Question: *Is I a $\mathcal{G}$-repair of H?*

PROBLEM REPAIR-EXISTENCE$(\mathcal{G}, b)$
Input: *A weighted hypergraph H in $\mathbf{WH}[b]$; a rational number q.*
Question: *Does H have a $\mathcal{G}$-repair I such that $\mathcal{G}_{[w]}(I) \geq q$?*

PROBLEM SUITABILITY-CHECKING$(\mathcal{G}, b)$
Input: *A weighted hypergraph H in $\mathbf{WH}[b]$; a set I of vertices; a rational number q.*
Question: *Is I a q-suitable set of H (with respect to $\mathcal{G}$)?*

These problems obviously have relationships among them. For example, if the answer to SUITABILITY-CHECKING$(\mathcal{G}, b)$ on input H, I, q is "yes," then the answer to REPAIR-EXISTENCE$(\mathcal{G}, b)$ on input H, q is also "yes." Also, for a weighted hypergraph H, if $q \coloneqq \max\{\mathcal{G}_{[w]}(J) \mid J \text{ is an independent set of } H\}$, then every $\mathcal{G}$-repair is a q-suitable set, and vice versa. We now give some computational complexity results. The proof of the following result is straightforward.

Theorem 1 (Complexity upper bounds). *For every $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}$ and $b \geq 2$,* REPAIR-CHECKING$(\mathcal{G}, b)$*, and* SUITABILITY-CHECKING$(\mathcal{G}, b)$ *are in* **coNP***, and* REPAIR-EXISTENCE$(\mathcal{G}, b)$ *is in* **NP***.*

The following result means that our problems are already intractable under the simplest parametrization.

Theorem 2 (Complexity lower bounds). REPAIR-CHECKING$(\mathsf{COUNT}, 2)$ *is* **coNP***-hard and* REPAIR-EXISTENCE$(\mathsf{COUNT}, 2)$ *is* **NP***-hard.*

Proof. The following is a well-known **NP**-complete problem [14]:

PROBLEM: INDEPENDENT SET
Input: A simple graph $G = (V, E)$; a positive integer $k \leq |V|$.
Question: Does G have an independent set I with cardinality $|I| \geq k$?

This problem is also referenced as MAX INDEPENDENT SET in the literature. There is a straightforward polynomial-time many-one reduction from the problem INDEPENDENT SET to REPAIR-EXISTENCE(COUNT, 2). We show next a polynomial-time many-one reduction from INDEPENDENT SET to the complement of REPAIR-CHECKING(COUNT, 2). Let $G = (V, E)$, k be an input to INDEPENDENT SET. Let I be a set of fresh vertices such that $|I| = k - 1$. Let F be the set of all edges $\{u, v\}$ such that $u \in I$ and $v \in V$. Clearly, I is an inclusion-maximal independent set of the graph $H := (V \cup I, E \cup F)$, and the pair H, I is a legal input to REPAIR-CHECKING(COUNT, 2). It is now easily verified that G has an independent set of cardinality $\geq k$ if and only if I is not a COUNT-repair of H. This concludes the proof. □

On the other hand, SUITABILITY-CHECKING(COUNT, 2) is tractable (i.e., in **P**). Indeed, tractability holds for a larger class of aggregation functions that contains COUNT and is defined next.

Definition 7 ($\subseteq$-monotone). *An aggregation function is called* $\subseteq$-monotone *if for every weighted set (I, w), for all $J_1, J_2 \subseteq I$ such that $J_1 \subseteq J_2$, it holds that $\mathcal{G}_{[w]}(J_1) \leq \mathcal{G}_{[w]}(J_2)$.*[1]

It is easily verified that COUNT and MAX are $\subseteq$-monotone. SUM is also $\subseteq$-monotone, because we do not consider negative numbers. On the other hand, MIN and AVG are not $\subseteq$-monotone. We give the following claim without proof, because we will shortly prove a stronger result.

Claim (Complexity upper bound). For every $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}$ and $b \geq 2$, if $\mathcal{G}$ is $\subseteq$-monotone, then SUITABILITY-CHECKING$(\mathcal{G}, b)$ is in **P**.

5 Main Tractability Theorem

Theorem 2 shows that REPAIR-CHECKING$(\mathcal{G}, b)$ becomes already intractable for simple aggregation functions and conflict hypergraphs. The aim of the current section is to better understand the reason for this intractability, and to identify aggregation functions for which REPAIR-CHECKING$(\mathcal{G}, b)$ is tractable. In Sections 5.1 and 5.2, we define two properties of aggregation functions that give rise to some first tractability results. Then, in Section 5.3, we combine these results in our main tractability theorem for REPAIR-CHECKING$(\mathcal{G}, b)$.

[1] In the notation $\mathcal{G}_{[w]}(J_1)$, the weight function is understood to be the restriction of w to J_1.

5.1 Monotone Under Priority

The converse of the claim at the end of Section 4 does not hold. Indeed, MIN is not $\subseteq$-monotone, but it is easily verified that SUITABILITY-CHECKING(MIN, b) is in **P**. We now aim at larger classes of aggregation functions $\mathcal{G}$ for which SUITABILITY-CHECKING($\mathcal{G}, b$) is in **P**. The computational complexity of this problem is mainly incurred by the saturation property in Definition 5, as there can be exponentially many sets including a given independent set. Therefore, we are looking for conditions that avoid such an exponential search. Such a condition is given in Definition 8.

Definition 8 (Monotone under priority). *We say that an aggregation function $\mathcal{G}$ is* monotone under priority *if for every weighted set V, for every $I \subseteq V$, it is possible to compute, in polynomial time in $|V|$, a set $S \subseteq V \setminus I$ whose powerset 2^S contains all and only those subsets of $V \setminus I$ that can be unioned with I without incurring a decrease of the aggregated value (i.e., for every $J \subseteq V \setminus I$, the following holds true: $J \subseteq S$ if and only if $\mathcal{G}_{[w]}(I) \leq \mathcal{G}_{[w]}(I \cup J)$).*

We write $\mathbf{AGG}^{\mathsf{poly}}_{\mathsf{mon}}$ for the set of aggregation functions in $\mathbf{AGG}^{\mathsf{poly}}$ that are monotone under priority.

To illustrate Definition 8, we show that MIN is monotone under priority. To this end, let V be a weighted set and $I \subseteq V$. Clearly, $\mathsf{MIN}_{[w]}(I) \leq \mathsf{MIN}_{[w]}(I \cup J)$ if (and only if) J contains no element with weight strictly smaller than $\mathsf{MIN}_{[w]}(I)$. Therefore, the set $S = \{v \in V \setminus I \mid w(v) \geq \mathsf{MIN}_{[w]}(I)\}$ shows that MIN is monotone under priority. It is even easier to show that every $\subseteq$-monotone aggregation function in $\mathbf{AGG}^{\mathsf{poly}}$ is monotone under priority, by letting $S = V \setminus I$. Therefore, the following lemma is more general than the claim at the end of Section 4.

Lemma 1. *For every $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}_{\mathsf{mon}}$ and $b \geq 2$,* SUITABILITY-CHECKING($\mathcal{G}, b$) *is in* **P**.

Proof. Let $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}$ be a function that is monotone under priority. Let H, I, q be an input to SUITABILITY-CHECKING($\mathcal{G}, b$). If $\mathcal{G}_{[w]}(I) < q$ or I is not an independent set, return "no"; otherwise the *saturation* condition in the definition of q-suitable sets remains to be verified. To this end, compute in polynomial time the set S mentioned in Definition 8. Then compute in polynomial time its subset $S' := \{v \in S \mid I \cup \{v\} \text{ is an independent set}\}$. By Definition 5, I is saturated (and hence q-suitable) if and only if there is no nonempty set $J \subseteq V \setminus I$ such that $I \cup J$ is independent and $\mathcal{G}_{[w]}(I) \leq \mathcal{G}_{[w]}(I \cup J)$. Consequently, by Definition 8, I is saturated if and only if $S' = \emptyset$, which can be tested in polynomial time. □

Among the five common aggregation functions COUNT, SUM, MAX, MIN, and AVG, the latter one is the only one that is not in $\mathbf{AGG}^{\mathsf{poly}}_{\mathsf{mon}}$, as illustrated next.

Example 1. We show that the aggregation function AVG is not monotone under priority. Let $V = \{a, b, c, d\}$. Let $w : V \to \{1, 2\}$ such that $w(a) = w(b) = 1$ and $w(c) = w(d) = 2$. Let $I = \{a, c\}$. Then, $\mathsf{AVG}_{[w]}(I) = \frac{3}{2}$. The subsets of

$V \setminus I = \{b, d\}$ that can be unioned with I without incurring a decrease of AVG are $\{\}$, $\{d\}$, and $\{b, d\}$. However, the set of the latter three sets is not the powerset of some other set.

5.2 k-Combinatorial

Lemma 1 tells us that SUITABILITY-CHECKING$(\mathcal{G}, b)$ is in **P** if $\mathcal{G} =$ MIN or $\mathcal{G} =$ MAX. However, an easier explanation is that the aggregated values of MIN and MAX over a weighted set are determined by a single element in that set. This observation motivates the following definition.

Definition 9 (k-combinatorial). *Let k be a positive integer. We say that an aggregation function $\mathcal{G}$ is* k-combinatorial *if every weighted set I includes a subset J such that $|J| \leq k$ and $\mathcal{G}_{[w]}(J) = \mathcal{G}_{[w]}(I)$. If $\mathcal{G}$ is not k-combinatorial for any k, we say that $\mathcal{G}$ is* full-combinatorial.

We write $\mathbf{AGG}^{\mathsf{poly}}_{(k)}$ for the set of aggregation functions in $\mathbf{AGG}^{\mathsf{poly}}$ that are k-combinatorial.

Obviously, the aggregation functions MIN and MAX are 1-combinatorial. From this, we can easily obtain an aggregation function that is 2-combinatorial. For example, define SPREAD as the aggregation function such that for every weighted set I, $\mathsf{SPREAD}_{[w]}(I) := \mathsf{MAX}_{[w]}(I) - \mathsf{MIN}_{[w]}(I)$. The notion of k-combinatorial also naturally relates to the well-studied notion of top-k queries. For example, for a fixed k and an aggregation function $\mathcal{G}$, we can define a new aggregation function that, on input of any weighted set (I, w), returns $\max\{\mathcal{G}_{[w]}(J) \mid J \subseteq I, |J| = k\}$, i.e., the highest $\mathcal{G}$-value found in any subset of size exactly k (and returns 0 if $|I| < k$). This new aggregation function is k-combinatorial by construction.

Lemma 2. *Let k be a positive integer. For every $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}_{(k)}$ and $b \geq 2$,* REPAIR-EXISTENCE$(\mathcal{G}, b)$ *is in* **P**.

Proof. Let H, q be an input to REPAIR-EXISTENCE$(\mathcal{G}, b)$. We can compute in polynomial time the value m defined as follows:

$$m := \max\{\mathcal{G}_{[w]}(J) \mid J \text{ is an independent set of } H \text{ with } |J| \leq k\}. \tag{1}$$

Since $\mathcal{G}$ is k-combinatorial, every repair I of H satisfies $\mathcal{G}_{[w]}(I) = m$. Thus, the answer to REPAIR-EXISTENCE$(\mathcal{G}, b)$ is "yes" if $q = m$, and "no" otherwise. □

5.3 Main Tractability Theorem

By bringing together the results of the two preceding subsections, we obtain our main tractability result.

Theorem 3 (Main tractability theorem). *Let k be a positive integer. For every $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}_{(k)} \cap \mathbf{AGG}^{\mathsf{poly}}_{\mathsf{mon}}$, for every $b \geq 2$,* REPAIR-CHECKING$(\mathcal{G}, b)$ *is in* **P**.

Proof. Let $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}_{(k)} \cap \mathbf{AGG}^{\mathsf{poly}}_{\mathsf{mon}}$. Let H, I be an input to the problem $\mathsf{REPAIR\text{-}CHECKING}(\mathcal{G}, b)$. We can compute, in polynomial time, the value m defined by (1) in the proof of Lemma 2. If $\mathcal{G}_{[w]}(I) < m$, return "no"; otherwise we solve $\mathsf{SUITABILITY\text{-}CHECKING}(\mathcal{G}, b)$ with input H, I, m, which is in $\mathbf{P}$ by Lemma 1. In particular, if H, I, m is a "no"-instance of the problem $\mathsf{SUITABILITY\text{-}CHECKING}(\mathcal{G}, b)$, return "no". If we have not answered "no" so far, then $\mathcal{G}_{[w]}(I) = m$ and I is an m-suitable set of H; in this case, return "yes". It is clear that this decision procedure is correct and runs in polynomial time. □

6 On Full-Combinatorial Aggregation Functions

Lemma 2 stated that the problem $\mathsf{REPAIR\text{-}EXISTENCE}(\mathcal{G}, b)$ is tractable if $\mathcal{G}$ is k-combinatorial for some fixed k. We will now show that dropping this condition quickly results in intractability. For a technical reason that will become apparent in the proof of Theorem 4, we need the following definition.

Definition 10 (witnessable). *Let $\mathcal{G}$ be an aggregation function that is full-combinatorial. We say that $\mathcal{G}$ is* witnessable *if the following task is in polynomial time:*

Input: *A positive integer k in unary. That is, a string $111\cdots1$ of length k.*
Output: *Return a shortest sequence $(q_1, q_2, \ldots, q_n)$ of nonnegative rational numbers witnessing that $\mathcal{G}$ is not k-combinatorial ($n > k$). Formally, for the weight function w that maps each i to q_i ($1 \leq i \leq n$), it must hold that for every $N \subseteq \{1, 2, \ldots, n\}$ with $|N| \leq k$, we have $\mathcal{G}_{[w]}(N) \neq \mathcal{G}_{[w]}(\{1, 2, \ldots, n\})$.*

Clearly, if $\mathcal{G}$ is full-combinatorial, the output requested in Definition 10 exists for every k. Therefore, the crux is that the definition asks to return such an output in polynomial time, where it is to be noted that the input is encoded in unary, i.e., has length k (and not $\log k$). Since aggregation functions $\mathcal{G}$ are closed under isomorphism, any permutation of a valid output is still a valid output. An example of a witnessable aggregation function is SUM: on input k, a valid output is the sequence $(1, 1, \ldots, 1)$ of length $k + 1$. For full-combinatorial functions in $\mathbf{AGG}^{\mathsf{poly}}$, the requirement of being witnessable seems very mild, and is expected to be fulfilled by natural aggregation functions.

The following result generalizes a complexity lower bound previously established by Theorem 2, because COUNT obviously satisfies the conditions imposed on $\mathcal{G}$ in the following theorem statement.

Theorem 4. *Let $\mathcal{G} \in \mathbf{AGG}^{\mathsf{poly}}$ be a full-combinatorial function that is $\subseteq$-monotone and witnessable. Then $\mathsf{REPAIR\text{-}EXISTENCE}(\mathcal{G}, 2)$ is $\mathbf{NP}$-complete.*

Proof. Membership in $\mathbf{NP}$ follows from Theorem 1. The $\mathbf{NP}$-hardness proof is a polynomial-time many-one reduction from $\mathsf{3SAT}$. To this end, let φ be an instance of $\mathsf{3SAT}$ with k clauses. Let $(q_1, q_2, \ldots, q_n)$ with $n > k$ be the output of the task defined in Definition 10. Let w be the weight function that maps

each i to q_i ($1 \leq i \leq n$), and let $Q \coloneqq \mathcal{G}_{[w]}(\{1, \ldots, n\})$. Assume for the sake of contradiction that for some strict subset N of $\{1, \ldots, n\}$, we have $\mathcal{G}_{[w]}(N) = Q$. Since $\mathcal{G}$ is k-combinatorial, $|N| \geq k + 1$. Then the sequence $(q_i)_{i \in N}$ of length $< n$ witnesses that $\mathcal{G}$ is not k-combinatorial, contradicting that Definition 10 requires a shortest witness. We conclude by contradiction that $N \subsetneq \{1, 2, \ldots, n\}$ implies $\mathcal{G}_{[w]}(N) \neq Q$. Since $\mathcal{G}$ is $\subseteq$-monotone, it follows that $N \subsetneq \{1, 2, \ldots, n\}$ implies $\mathcal{G}_{[w]}(N) < Q$.

The reduction constructs, in polynomial time in the length of φ, a weighted graph $H = ((V, w'), E)$ as follows. If the ith clause of φ is $\ell_1 \vee \ell_2 \vee \ell_3$, where ℓ_1, ℓ_2, ℓ_3 are positive or negative literals, then (i, ℓ_1), (i, ℓ_2), (i, ℓ_3) are vertices of V that form a triangle in E, and these three vertices are mapped to q_i by w'. For every propositional variable p, if (i, p) and $(j, \neg p)$ are vertices, then they are connected by an edge. Finally, we add isolated fresh vertices v_{k+1}, v_{k+2}, …, v_n, and let $w'(v_j) = q_j$ for $k+1 \leq j \leq n$. We claim that the following are equivalent:

1. φ has a satisfying truth assignment; and
2. H has a $\mathcal{G}$-repair I such that $\mathcal{G}_{[w']}(I) \geq Q$.

For the direction 1 $\implies$ 2, let τ be a satisfying truth assignment for φ. Construct I as follows. First, I includes $\{v_{k+1}, v_{k+2}, \ldots, v_n\}$. Then, for i ranging from 1 to the number k of clauses, if the ith clause of φ is $\ell_1 \vee \ell_2 \vee \ell_3$, we pick $g \in \{1, 2, 3\}$ such that ℓ_g evaluates to true under τ, and add (i, ℓ_g) to I. In this way, I contains exactly one vertex from each triangle. Moreover, since τ is a truth assignment, we will never insert into I both (i, p) and $(j, \neg p)$ for a same propositional variable p. By construction, I is an independent set of H containing n elements, and $\mathcal{G}_{[w']}(I) = \mathcal{G}_{[w]}(\{1, \ldots, n\}) = Q$.

For the direction 2 $\implies$ 1, let I be a $\mathcal{G}$-repair such that $\mathcal{G}_{[w']}(I) \geq Q$. Then, from our construction of H and our previous result that Q can only be attained if all q_is are aggregated, it follows that for every $i \in \{1, \ldots, k\}$, there is a literal ℓ in the ith clause such that I contains the vertex (i, ℓ). Moreover, since I is an independent set, it cannot contain both (i, p) and $(j, \neg p)$ for a same propositional variable p. Then I obviously defines a satisfying truth assignment for φ. This concludes the proof. □

7 Conclusion and Future Work

Our work combines and generalizes notions from databases and graph theory. From a database-theoretical viewpoint, $\mathcal{G}$-repairs extend subset- and cardinality-repairs by allowing arbitrary aggregation functions. From a graph-theoretical viewpoint, $\mathcal{G}$-repairs extend maximum weighted independent sets by allowing hypergraphs as well as aggregation functions other than SUM. With minor effort, complexity lower bounds for REPAIR-CHECKING$(\mathcal{G}, b)$ were obtained from known results about maximum (weighted) independent sets. Our main result is the computational tractability result of Theorem 3, which shows a polynomial upper time bound on this problem under some restrictions that are not unrealistic (and are actually met by several common aggregation functions).

Throughout this paper, aggregation functions and their properties were defined and treated in an abstract, semantic way. In the future, we want to study logical languages that allow expressing aggregation functions (e.g., first-order logic with aggregation), and in particular their syntactic restrictions that guarantee tractability.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. F. N. Afrati and P. G. Kolaitis. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 31–41. ACM, 2009.
3. M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.
4. F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.
5. L. E. Bertossi. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.
6. L. E. Bertossi. Database repairs and consistent query answering: Origins and further developments. In *PODS*, pages 48–58. ACM, 2019.
7. L. E. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Inf. Syst.*, 33(4-5):407–434, 2008.
8. P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD Conference*, pages 143–154. ACM, 2005.
9. J. Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2007.
10. J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.*, 197(1-2):90–121, 2005.
11. J. Chomicki, J. Marcinkowski, and S. Staworko. Computing consistent query answers using conflict hypergraphs. In *CIKM*, pages 417–426. ACM, 2004.
12. J. Du, G. Qi, and Y. Shen. Weight-based consistent query answering over inconsistent $\mathcal{SHIQ}$ knowledge bases. *Knowl. Inf. Syst.*, 34(2):335–371, 2013.
13. S. Flesca, F. Furfaro, and F. Parisi. Querying and repairing inconsistent numerical databases. *ACM Trans. Database Syst.*, 35(2):14:1–14:50, 2010.
14. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
15. L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.
16. D. Maslowski and J. Wijsen. Uncertainty that counts. In *FQAS*, volume 7022 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2011.
17. S. Staworko and J. Chomicki. Consistent query answers in the presence of universal constraints. *Inf. Syst.*, 35(1):1–22, 2010.
18. J. Wijsen. Foundations of query answering on inconsistent databases. *SIGMOD Rec.*, 48(3):6–16, 2019.