# Computing Optimal Repairs for Functional Dependencies

Ester Livshits
Technion
Haifa, Israel
esterliv@cs.technion.ac.il

Benny Kimelfeld
Technion
Haifa, Israel
bennyk@cs.technion.ac.il

Sudeepa Roy
Duke University
Durham, NC, USA
sudeepa@cs.duke.edu

## ABSTRACT

We investigate the complexity of computing an optimal repair of an inconsistent database, in the case where integrity constraints are Functional Dependencies (FDs). We focus on two types of repairs: an optimal subset repair (optimal S-repair) that is obtained by a minimum number of tuple deletions, and an optimal update repair (optimal U-repair) that is obtained by a minimum number of value (cell) updates. For computing an optimal S-repair, we present a polynomial-time algorithm that succeeds on certain sets of FDs and fails on others. We prove the following about the algorithm. When it succeeds, it can also incorporate weighted tuples and duplicate tuples. When it fails, the problem is NP-hard, and in fact, APX-complete (hence, cannot be approximated better than some constant). Thus, we establish a dichotomy in the complexity of computing an optimal S-repair. We present general analysis techniques for the complexity of computing an optimal U-repair, some based on the dichotomy for S-repairs. We also draw a connection to a past dichotomy in the complexity of finding a "most probable database" that satisfies a set of FDs with a single attribute on the left hand side; the case of general FDs was left open, and we show how our dichotomy provides the missing generalization and thereby settles the open problem.

## KEYWORDS

Inconsistent Databases; Database Cleaning; Optimal Repairs; Cardinality Repairs; Value Repairs; Functional Dependencies; Dichotomy; Approximation

**ACM Reference Format:**
Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2018. Computing Optimal Repairs for Functional Dependencies. In *PODS'18: 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3196959.3196980

## 1 INTRODUCTION

Database inconsistency arises in a variety of scenarios and for different reasons. Data may be collected from imprecise sources (social encyclopedias/networks, sensors attached to appliances, cameras, etc.) via imprecise procedures (natural-language processing, signal processing, image analysis, etc.). Inconsistency may also arise when integrating databases of different organizations with conflicting information, or even consistent information in conflicting formats. Arenas et al. [5] introduced a principled approach to managing inconsistency via the notions of *repairs* and *consistent query answering*. An *inconsistent database* is a database $D$ that violates integrity constraints, a *repair* is a consistent database $D'$ obtained from $D$ by a minimal sequence of operations, and the *consistent answers* to a query are the answers given in every repair $D'$.

Instantiations of the repair framework differ in their definitions of *integrity constraints*, *operations*, and *minimality* [1]. Common types of constraints are denial constraints [18] that include the classic functional dependencies (FDs), and inclusion dependencies [11] that include the referential (foreign-key) constraints. An operation can be a *deletion* of a tuple, an *insertion* of a tuple, and an *update* of an attribute (cell) value. Minimality can be either *local*—no strict subset of the operations achieves consistency, or *global*—no smaller (or cheaper) subset achieves consistency. For example, if only tuple deletions are allowed, then a *subset repair* [12] corresponds to a local minimum (restoring any deleted tuple causes inconsistency) and a *cardinality repair* [29] corresponds to a global minimum (consistency cannot be gained by fewer tuple deletions). The *cost* of operations may differ between tuples; this can represent different levels of trust that we have in the tuples [24, 29].

In this paper, we focus on global minima under FDs via tuple deletions and value updates. Each tuple is associated with a weight that determines the cost of its deletion or a change of a single value. We study the complexity of computing a minimum repair in two settings: *(a)* only tuple deletions are allowed, that is, we seek a (weighted) cardinality repair, *and (b)* only value updates are allowed, that is, we seek what Kolahi and Lakshmanan [24] refer to as an "optimum V-repair." We refer to the two challenges as computing an optimal subset repair (optimal S-repair) and computing an optimal update repair (optimal U-repair).

The importance of computing an optimal repair arises in the challenge of *data cleaning* [17]—eliminate errors and dirt (manifested as inconsistencies) from the database. Specifically, our motivation is twofold. The obvious motivation is in fully automated cleaning, where an optimal repair is the best candidate, assuming the system is aware of only the constraints and tuple weights. The second motivation comes from the more realistic practice of iterative, human-in-the-loop cleaning [6, 9, 13, 19]. There, the cost of the optimal repair can serve as an educated estimate for the extent to which the database is dirty and, consequently, the amount of effort needed for completion of cleaning.

As our integrity constraints are FDs, it suffices to consider a database with a single relation, which we call here a *table*. In a general database, our results can be applied to each relation individually. A table $T$ conforms to a relational schema $R(A_1, \ldots, A_k)$ where each $A_i$ is an attribute. Integrity is determined by a set $\Delta$ of FDs. Our


*PODS'18, June 10–15, 2018, Houston, TX, USA*

ACM ISBN 978-1-4503-4706-8/18/06…$15.00
https://doi.org/10.1145/3196959.3196980

complexity analysis focuses primarily on *data complexity*, where $R(A_1, \ldots, A_k)$ and $\Delta$ are considered fixed and only $T$ is considered input. Hence, we have infinitely many optimization problems, one for each combination of $R(A_1, \ldots, A_k)$ and $\Delta$. Table records have identifiers, as we wish to be able to determine easily which cells are updated in a repair. Consequently, we allow duplicate tuples (with distinct identifiers).

We begin with the problem of computing an optimal S-repair. The problem is known to be computationally hard for denial constraints [29]. As we discuss later, complexity results can be inferred from prior work [20] for FDs with a single attribute on the left hand side (lhs for short). For general FDs, we present the algorithm OptSRepair (Algorithm 1). The algorithm seeks opportunities for simplifying the problem by eliminating attributes and FDs, until no FDs are left (and then the problem is trivial). For example, if all FDs share an attribute $A$ on the left hand side, then we can partition the table according to $A$ and solve the problem separately on each partition; but now, we can ignore $A$. We refer to this simplification as "common lhs." Two additional simplifications are the "consensus" and "lhs marriage." Importantly, the algorithm terminates in polynomial time, even under *combined* complexity.

However, OptSRepair may fail by reaching a nonempty set of FDs where no simplification can be applied. We prove two properties of the algorithm. The first is *soundness*—if the algorithm succeeds, then it returns an optimal S-repair. More interesting is the property of *completeness*—if the algorithm fails, then the problem is NP-hard. In fact, in this case the problem is APX-complete, that is, for some $\epsilon > 0$ it is NP-hard to find a consistent subset with a cost lower than $(1 + \epsilon)$ times the minimum, but *some* $(1 + \epsilon')$ is achievable in polynomial time. More so, the problem remains APX-complete if we assume that the table does not contain duplicates, and all tuples have a unit weight (in which case we say that $T$ is *unweighted*). Consequently, we establish the following dichotomy in complexity for the space of combinations of schemas $R(A_1, \ldots, A_k)$ and FD sets $\Delta$.

- If we can eliminate all nontrivial FDs in $\Delta$ with the three simplifications, then an optimal S-repair can be computed in polynomial time using OptSRepair.
- Otherwise, computing an optimal S-repair is APX-complete, even for unweighted tables without duplicates.

We then continue to the problem of computing an optimal U-repair. Here we do not establish a full dichotomy, but we make a substantial progress. We have found that proving hardness results for updates is far more subtle than for deletions. We identify conditions where the complexity of computing an optimal U-repair and that of computing an optimal S-repair coincide. One such condition is the common lhs (i.e., all FDs share a left-hand-side attribute). Hence, in this case, our dichotomy provides the precise test of tractability. We also show decomposition techniques that extend the opportunities of using the dichotomy. As an example, consider Purchase(product, price, buyer, email, address) and $\Delta_0 = \{\text{product} \rightarrow \text{price}, \text{buyer} \rightarrow \text{email}\}$. We can decompose this problem into $\Delta_1 = \{\text{product} \rightarrow \text{price}\}$ and $\Delta_2 = \{\text{buyer} \rightarrow \text{email}\}$, and consider each $\Delta_i$, for $i = 1, 2$, independently. The complexity of each $\Delta_i$ is the same in both variants of optimal repairs, and so, polynomial time. Yet, these results do not cover all sets of FDs. For example, let $\Delta_3 = \{\text{email} \rightarrow \text{buyer}, \text{buyer} \rightarrow \text{address}\}$. Kolahi and Lakshmanan [24] proved that under $\Delta_3$, computing an optimal U-repair is NP-hard. Our dichotomy shows that it is also NP-hard (and also APX-complete) to compute an S-repair under $\Delta_3$. Yet, this FD set does not fall in our coincidence cases.

The above defined $\Delta_0$ is an example where an optimal U-repair can be computed in polynomial time, but computing an optimal S-repair is APX-complete. We also show an example in the reverse direction, namely $\Delta_4 = \{\text{buyer} \rightarrow \text{email}, \text{email} \rightarrow \text{buyer}, \text{buyer} \rightarrow \text{address}\}$. This FD set falls in the positive side of our dichotomy for optimal S-repairs, but computing an optimal U-repair is APX-complete. The proof of APX-hardness is inspired by, but considerably more involved than, the hardness proof of Kolahi and Lakshmanan [24] for $\Delta_3$.

Finally, we consider approximate repairing. For the case of an optimal S-repair, the problem easily reduces to that of *weighted vertex cover*, and hence, we get a polynomial-time 2-approximation due to Bar-Yehuda and Even [7]. To approximate optimal U-repairs, we show an efficient reduction to S-repairs, where the loss in approximation is linear in the number of attributes. Hence, we obtain a constant-ratio approximation, where the constant has a linear dependence on the number of attributes. Kolahi and Lakshmanan [24] also gave an approximation for optimal U-repairs, but their worst-case approximation can be quadratic in the number of attributes. We show an infinite sequence of FD sets where this gap is actually realized. On the other hand, we also show an infinite sequence where our approximation is linear in the number of attributes, but theirs remains constant. Hence, in general, the two approximations are incomparable, and we can combine the two by running both approximations and taking the best.

Stepping outside the framework of repairs, a different approach to data cleaning is *probabilistic* [4, 20, 30]. The idea is to define a probability space over possible clean databases, where the probability of a database is determined by the extent to which it satisfies the integrity constraints. The goal is to find a *most probable database* that, in turn, serves as the clean outcome. As an instantiation, Gribkoff, Van den Broeck, and Suciu [20] identify probabilistic cleaning as the "Most Probable Database" problem (MPD): given a tuple-independent probabilistic database [14, 32] and a set of FDs, find the most probable database among those satisfying the FDs (or, put differently, condition the probability space on the FDs). They show a dichotomy for unary FDs (i.e., FDs with a single attribute on the left hand side). The case of general (not necessarily unary) FDs has been left open. It turns out that there are reductions from MPD to computing an optimal S-repair and vice versa. Consequently, we are able to generalize their dichotomy to all FDs, and hence, fully settle the open problem.

## 2 PRELIMINARIES

We first present some basic terminology and notation that we use throughout the paper.

### 2.1 Schemas and Tables

An instance of our data model is a single table where each tuple is associated with an *identifier* and a *weight* that states how costly it is to change or delete the tuple. Such a table corresponds to a

*relation schema* that we denote by $R(A_1, \ldots, A_k)$, where $R$ is the *relation name* and $A_1, \ldots, A_k$ are distinct *attributes*. We say that $R(A_1, \ldots, A_k)$ is *k-ary* since it has $k$ attributes. When there is no risk of confusion, we may refer to $R(A_1, \ldots, A_k)$ by simply $R$.

We use capital letters from the beginning of the English alphabet (e.g., $A, B, C$), possibly with subscripts and/or superscripts, to denote individual attributes, and capital letters from the end of the English alphabet (e.g., $X, Y, Z$), possibly with subscripts and/or superscripts, to denote sets of attributes. We follow the convention of avoiding commas and curly braces when writing sets of attributes (e.g., $ABC$).

We assume a countably infinite domain Val of attribute values. By a *tuple* we mean a sequence of values in Val. A *table* $T$ over $R(A_1, \ldots, A_k)$ has a collection $ids(T)$ of (tuple) identifiers and it maps every identifier $i$ to a tuple in $\text{Val}^k$ and a positive weight; we denote this tuple by $T[i]$ and this weight by $w_T(i)$. For $i \in ids(T)$ we refer to $T[i]$ as a tuple *of* $T$. We denote by $T[*]$ the set of all tuples of $T$. We say that $T$ is:

- *duplicate free* if distinct tuples disagree on at least one attribute, that is, $T[i] \neq T[j]$ whenever $i \neq j$;
- *unweighted* if all tuple weights are equal, that is, $w_T(i) = w_T(j)$ for all identifiers $i$ and $j$.

We use $|T|$ to denote the number of tuple identifiers of $T$, that is, $|T| \stackrel{\text{def}}{=} |ids(T)|$. Let $\mathbf{t} = (a_1, \ldots, a_k)$ be a tuple of $T$. We use $\mathbf{t}.A_j$ to refer to the value $a_j$. If $X = A_{i_1}, \ldots, A_{i_\ell}$ is a sequence of attributes in $\{A_1, \ldots, A_k\}$, then $\mathbf{t}[X]$ denotes the tuple $(\mathbf{t}.A_{i_1}, \ldots, \mathbf{t}.A_{i_\ell})$.

Example 2.1. Our running example is based on the tables of Figure 1, over the schema Office(facility, room, floor, city), describing the location of offices in an organization. For example, the tuple $T[1]$ corresponds to an office in room 322, in the third floor of the headquarters (HQ) building, located in Paris. The meaning of the yellow background color will be clarified later. The identifier of each tuple is shown on the leftmost (gray shaded) column, and its weight on the rightmost column (also gray shaded). Note that table $S_2$ is duplicate free and unweighted, table $S_1$ is duplicate free but not unweighted, and table $U_2$ is neither duplicate free nor unweighted. □

## 2.2 Functional Dependencies (FDs)

Let $R(A_1, \ldots, A_k)$ be a schema. As usual, an FD (over $R$) is an expression of the form $X \to Y$ where $X$ and $Y$ are sequences of attributes of $R$. We refer to $X$ as the *left-hand side*, or *lhs* for short, and to $Y$ as the *right-hand side*, of *rhs* for short. A table $T$ *satisfies* $X \to Y$ if every two tuples that agree on $X$ also agree on $Y$; that is, for all $\mathbf{t}, \mathbf{s} \in T[*]$, if $\mathbf{t}[X] = \mathbf{s}[X]$ then $\mathbf{t}[Y] = \mathbf{s}[Y]$. We say that $T$ *satisfies* a set $\Delta$ of FDs if $T$ satisfies each FD in $\Delta$; otherwise, $T$ *violates* $\Delta$.

An FD $X \to Y$ is *entailed by* $\Delta$, denoted $\Delta \models X \to Y$, if every table $T$ that satisfies $\Delta$ also satisfies the FD $X \to Y$. The *closure* of $\Delta$, denoted $cl(\Delta)$, is the set of all FDs over $R$ that are entailed by $\Delta$. The *closure* of an attribute set $X$ (*w.r.t.* $\Delta$), denoted $cl_\Delta(X)$, is the set of all attributes $A$ such that the FD $X \to A$ is entailed by $\Delta$. Two sets $\Delta_1$ and $\Delta_2$ of FDs are *equivalent* if they have the same closure (or in other words, each FD in $\Delta_1$ is entailed by $\Delta_2$ and vice versa, or put differently, every table that satisfies one also satisfies the other). An FD $X \to Y$ is *trivial* if $Y \subseteq X$; otherwise, it is *nontrivial*. Note that a trivial FD belongs to the closure of every set of FDs (including the empty one). We say that $\Delta$ is *trivial* if $\Delta$ does not contain any nontrivial FDs (e.g., it is empty); otherwise, $\Delta$ is *nontrivial*.

Next, we give some non-standard notation that we need for this paper. A *common lhs* of an FD set $\Delta$ is an attribute $A$ such that $A \in X$ for all FDs $X \to Y$ in $\Delta$. An FD set $\Delta$ is a *chain* if for every two FDs $X_1 \to Y_1$ and $X_2 \to Y_2$ it is the case that $X_1 \subseteq X_2$ or

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 1 | HQ | 322 | 3 | Paris | 2 |
| 2 | HQ | 322 | 30 | Madrid | 1 |
| 3 | HQ | 122 | 1 | Madrid | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(a) Table $T$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 2 | HQ | 322 | 30 | Madrid | 1 |
| 3 | HQ | 122 | 1 | Madrid | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(b) Consistent subset $S_1$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 1 | HQ | 322 | 3 | Paris | 2 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(c) Consistent subset $S_2$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 3 | HQ | 122 | 1 | Madrid | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(d) Consistent subset $S_3$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 1 | F01 | 322 | 3 | Paris | 2 |
| 2 | HQ | 322 | 30 | Madrid | 1 |
| 3 | HQ | 122 | 1 | Madrid | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(e) Consistent update $U_1$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 1 | HQ | 322 | 3 | Paris | 2 |
| 2 | HQ | 322 | 3 | Paris | 1 |
| 3 | HQ | 122 | 1 | Paris | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(f) Consistent update $U_2$

| *id* | facility | room | floor | city | *w* |
|---|---|---|---|---|---|
| 1 | HQ | 322 | 30 | Madrid | 2 |
| 2 | HQ | 322 | 30 | Madrid | 1 |
| 3 | HQ | 122 | 1 | Madrid | 1 |
| 4 | Lab1 | B35 | 3 | London | 2 |

(g) Consistent update $U_3$

**Figure 1: For Office(facility, room, floor, city) and FDs facility → city and facility room → floor, a table $T$, consistent subsets $S_1$, $S_2$ and $S_3$, and consistent updates $U_1$, $U_2$ and $U_3$. Changed values are marked in yellow.**

$X_2 \subseteq X_1$. Livshits and Kimelfeld [27] proved that the class of chain FD sets consists of precisely the FD sets in which the *subset repairs*, which we define in Section 2.3, can be counted in polynomial time (assuming P ≠ #P). The chain FD sets will arise in this work as well.

Example 2.2. In our running example (Figure 1) the set $\Delta$ consists of the following FDs:

- facility → city: a facility belongs to a single city.
- facility room → floor: a room in a facility does not go beyond one floor.

Note that the FDs allow for the same room number to occur in different facilities (possibly on different floors, in different cities). The attribute facility is a common lhs. Moreover, $\Delta$ is a chain FD set, since {facility} ⊆ {facility, room}. Table $T$ (Figure 1(a)) violates $\Delta$, and the other tables (Figures 1(b)–1(g)) satisfy $\Delta$. □

An FD $X \rightarrow Y$ might be such that $X$ is empty, and then we denote it by $\emptyset \rightarrow Y$ and call it a *consensus* FD. Satisfying the consensus FD $\emptyset \rightarrow Y$ means that all tuples agree on $Y$, or in other words, the column that corresponds to each attribute in $Y$ consists of copies of the same value. For example, $\emptyset \rightarrow$ city means that all tuples have the same city. A *consensus attribute* (*of* $\Delta$) is an attribute in $cl_\Delta(\emptyset)$, that is, an attribute $A$ such that $\emptyset \rightarrow A$ is implied by $\Delta$. We say that $\Delta$ is *consensus free* if it has no consensus attributes.

## 2.3 Repairs

Let $R(A_1, \ldots, A_k)$ be a schema, and let $T$ be a table. A *subset* of $T$ is a table $S$ that is obtained from $T$ by eliminating tuples. More formally, table $S$ is a subset of $T$ if $ids(S) \subseteq ids(T)$ and for all $i \in ids(S)$ we have $S[i] = T[i]$ and $w_S(i) = w_T(i)$. If $S$ is a subset of $T$, then the *distance* from $S$ to $T$, denoted $dist_{\mathsf{sub}}(S, T)$, is the weighted sum of the tuples missing from $S$; that is,

$$dist_{\mathsf{sub}}(S, T) \stackrel{\text{def}}{=} \sum_{i \in ids(T) \setminus ids(S)} w_T(i)\,.$$

A *value update* of $T$ (or just *update* of $T$ for short) is a table $U$ that is obtained from $T$ by changing attribute values. More formally, a table $U$ is an update of $T$ if $ids(U) = ids(T)$ and for all $i \in ids(U)$ we have $w_U(i) = w_T(i)$. We adopt the definition of Kolahi and Lakshmanan [24] for the distance from $U$ to $T$. Specifically, if $\mathbf{u}$ and $\mathbf{t}$ are tuples of tables over $R$, then the *Hamming distance* $H(\mathbf{u}, \mathbf{t})$ is the number of attributes in which $\mathbf{u}$ and $\mathbf{t}$ disagree, that is, $H(\mathbf{u}, \mathbf{t}) = |\{j \mid \mathbf{u}.A_j \neq \mathbf{t}.A_j\}|$. If $U$ is an update of $T$ then the *distance* from $U$ to $T$, denoted $dist_{\mathsf{sub}}(U, T)$, is the weighted Hamming distance between $U$ and $T$ (where every changed value counts as the weight of the tuple); that is,

$$dist_{\mathsf{upd}}(U, T) \stackrel{\text{def}}{=} \sum_{i \in ids(T)} w_T(i) \cdot H(T[i], U[i])\,.$$

Let $R(A_1, \ldots, A_k)$ be a schema, let $T$ be table, and let $\Delta$ be a set of FDs. A *consistent subset* (*of* $T$ *w.r.t.* $\Delta$) is a subset $S$ of $T$ such that $S \models \Delta$, and a *consistent update* (*of* $T$ *w.r.t.* $\Delta$) is an update $U$ of $T$ such that $U \models \Delta$. A *subset repair*, or just *S-repair* for short, is a consistent subset that is not strictly contained in any other consistent subset. An *update repair*, or just *U-repair* for short, is a consistent update that becomes inconsistent if any set of updated values is restored to the original values in $T$. An *optimal subset repair* of $T$, or just *optimal S-repair* for short, is a consistent subset $S$ of $T$ such that $dist_{\mathsf{sub}}(S, T)$ is minimal among all consistent subsets of $T$. Similarly, an *optimal update repair* of $T$, or just *optimal U-repair* for short, is a consistent update $U$ of $T$ such that $dist_{\mathsf{upd}}(U, T)$ is minimal among all consistent updates of $T$. When there is risk of ambiguity, we may stress that the optimal S-repair (or U-repair) is *of* $T$ and *under* $\Delta$ or *under* $R$ *and* $\Delta$.

Every (S- or U-) optimal repair is a repair, but not necessarily vice versa. Clearly, a consistent subset (respectively, update) can be transformed into a (not necessarily optimal) S-repair (respectively, U-repair), with no increase of distance, in polynomial time. In fact, we do not really need the concept of a repair per se, and the definition is given mainly for compatibility with the literature (e.g., [1]). Therefore, unless explicitly stated otherwise, we do not distinguish between an S-repair and a consistent subset, and between a U-repair and a consistent update.

We also define *approximations* of optimal repairs in the obvious ways, as follows. For a number $\alpha \geq 1$, an $\alpha$-optimal S-repair is an S-repair $S$ of $T$ such that $dist_{\mathsf{sub}}(S, T) \leq \alpha\, dist_{\mathsf{sub}}(S', T)$ for all S-repairs $S'$ of $T$, and an $\alpha$-optimal U-repair is a U-repair $U$ of $T$ such that $dist_{\mathsf{upd}}(U, T) \leq \alpha\, dist_{\mathsf{upd}}(U', T)$ for all U-repairs $U'$ of $T$. In particular, an optimal S-repair (resp., optimal U-repair) is the same as a 1-optimal S-repair (resp., 1-optimal U-repair).

Example 2.3. In our running example (Figure 1), tables $S_1$, $S_2$ and $S_3$ are consistent subsets, and $U_1$, $U_2$ and $U_3$ are consistent updates. For clarity, we marked with yellow shading the values that were changed for constructing each $U_i$. We have $dist_{\mathsf{sub}}(S_1, T) = 2$ since the missing tuple (tuple 1) has the weight 2. We also have $dist_{\mathsf{sub}}(S_2, T) = 2$ and $dist_{\mathsf{sub}}(S_3, T) = 3$. The reader can verify that $S_1$ and $S_2$ are optimal S-repairs. Table $S_3$ is *not* an optimal S-repair since its distance to $T$ is greater than the minimum. Nevertheless, $S_3$ is an 1.5-optimal S-repair (since $3/2 = 1.5$). Similarly, we have $dist_{\mathsf{upd}}(U_1, T) = 2$, $dist_{\mathsf{upd}}(U_2, T) = 3$, and $dist_{\mathsf{upd}}(U_3, T) = 4$ (since $U_3$ is obtained by changing two values of a tuple of weight 2). □

It should be noted that the values of an update $U$ of a table $T$ are not necessarily taken from the *active domain* (i.e., values that occur in $T$). An example is the value `F01` of table $U_1$ in Figure 1(e). This has implications on the complexity of computing optimal U-repairs. We discuss a restriction on the allowed update values in Section 5.

## 2.4 Complexity

We adopt the conventional measure of *data complexity*, where the schema $R(A_1, \ldots, A_k)$ and dependency set $\Delta$ are assumed to be *fixed*, and only the table $T$ is considered *input*. In particular, a "polynomial" running time may have an exponential dependency on $k$, as in $O(|T|^k)$, Hence, each combination of $R(A_1, \ldots, A_k)$ and $\Delta$ defines a distinct problem of finding an optimal repair (of the relevant type), and different combinations may feature different computational complexities.

For the complexity of approximation, we use the following terminology. In an optimization problem $P$, each input $x$ has a space of solutions $y$, each associated with a cost $\mathrm{cost}(x, y)$. Given $x$, the goal is to compute a solution $y$ with a minimum cost. For $\alpha \geq 1$, an *$\alpha$-approximation* for $P$ is an algorithm that, for input $x$, produces an *$\alpha$-optimal* solution $y$, which means that $\mathrm{cost}(x, y) \leq \alpha \cdot \mathrm{cost}(x, y')$ for all solutions $y'$. The complexity class *APX* consists of all optimization problems that have a polynomial-time constant-factor

approximation. A polynomial-time reduction $f$ from an optimization problem $Q$ to an optimization problem $P$ is a *strict reduction* if for all $\alpha \geq 1$, any $\alpha$-optimal solution for $f(x)$ can be transformed in polynomial time into an $\alpha$-optimal solution for $x$ [25]; it is a *PTAS* (Polynomial-Time Approximation Scheme) reduction if for all $\alpha > 1$ there exists $\beta_\alpha > 1$ such that any $\beta_\alpha$-optimal solution for $f(x)$ can be transformed in polynomial time into an $\alpha$-optimal solution for $x$. A strict reduction is also a PTAS reduction, but not necessarily vice versa. A problem $P$ is *APX-hard* if there is a PTAS reduction to $P$ from every problem in APX; it is *APX-complete* if, in addition, it is in APX. If $P$ is APX-hard, then there is a constant $\alpha_P > 1$ such that $P$ cannot be approximated better than $\alpha_P$, or else P=NP.

## 3 COMPUTING AN OPTIMAL S-REPAIR

In this section, we study the problem of computing an optimal S-repair. We begin with some conventions.

*Assumptions and Notation.* Throughout this section we assume that every FD has a single attribute on its right-hand side, that is, it has the form $X \to A$. Clearly, this is not a limiting assumption, since replacing $X \to YZ$ with $X \to Y$ and $X \to Z$ preserves equivalence.

Let $\Delta$ be a set of FDs. If $X$ is a set of attributes, then we denote by $\Delta - X$ the set $\Delta'$ of FDs that is obtained from $\Delta$ by removing each attribute of $X$ from every lhs and rhs of every FD in $\Delta$. Hence, no attribute in $X$ occurs in $\Delta - X$. If $A$ is an attribute, then we may write $\Delta - A$ instead of $\Delta - \{A\}$.

An *lhs marriage* of an FD set $\Delta$ is a pair $(X_1, X_2)$ of distinct lhs of FDs in $\Delta$ with the following properties.

- $cl_\Delta(X_1) = cl_\Delta(X_2)$
- The lhs of every FD in $\Delta$ contains either $X_1$ or $X_2$ (or both).

Example 3.1. A simple example of an FD set with an lhs marriage is the following FD set.

$$\Delta_{A \leftrightarrow B \to C} \stackrel{\text{def}}{=} \{A \to B,\ B \to A,\ B \to C\} \tag{1}$$

As another example, consider the following FD set.

$$\Delta_1 \stackrel{\text{def}}{=} \{\mathsf{ssn} \to \mathsf{first},\ \mathsf{ssn} \to \mathsf{last},\ \mathsf{first\ last} \to \mathsf{ssn},\ \mathsf{ssn} \to \mathsf{address},\ \mathsf{ssn\ office} \to \mathsf{phone},\ \mathsf{ssn\ office} \to \mathsf{fax}\}$$

In $\Delta_1$ the pair $(\{\mathsf{ssn}\}, \{\mathsf{first}, \mathsf{last}\})$ is an lhs marriage. □

Finally, if $S$ is a subset of a table $T$, then we denote by $w_T(S)$ the sum of weights of the tuples of $S$, that is,

$$w_T(S) \stackrel{\text{def}}{=} \sum_{i \in ids(S)} w_T(i)$$

### 3.1 Algorithm

We now describe an algorithm for finding an optimal S-repair. The algorithm terminates in polynomial time, even under combined complexity, yet it may *fail*. If it succeeds, then the result is guaranteed to be an optimal S-repair. We later discuss the situations in which the algorithm fails. The algorithm, OptSRepair, is shown as Algorithm 1. The input is a set $\Delta$ of FDs and a table $T$, both over the same relation schema (that we do not need to refer to explicitly). In the remainder of this section, we fix $\Delta$ and $T$, and describe the execution of OptSRepair on $\Delta$ and $T$. In the pseudocode, we use conventional operators in relational algebra, namely projection ($\pi$), selection ($\sigma$) and union ($\cup$).

**Algorithm 1** OptSRepair$(\Delta, T)$

**if** $\Delta$ is trivial **then** ▷ *successful termination*
  **return** $T$
remove trivial FDs from $\Delta$
**if** $\Delta$ has a common lhs **then**
  **return** CommonLHSRep$(\Delta, T)$
**if** $\Delta$ has a consensus FD **then**
  **return** ConsensusRep$(\Delta, T)$
**if** $\Delta$ has an lhs marriage **then**
  **return** MarriageRep$(\Delta, T)$
**fail** ▷ *cannot find an optimal S-repair*

**Subroutine 1** CommonLHSRep$(\Delta, T)$

$A :=$ a common lhs of $\Delta$
**return** $\cup_{(a) \in \pi_A T[*]}$ OptSRepair$(\sigma_{A=a}T, \Delta - A)$

**Subroutine 2** ConsensusRep$(\Delta, T)$

select a consensus FD $\emptyset \to A$ in $\Delta$
**for all** $a \in \pi_A T[*]$ **do**
  $S_a :=$ OptSRepair$(\sigma_{A=a}T, \Delta - A)$
$a_{\max} := \operatorname{argmax}_a \{w_T(S_a) \mid (a) \in \pi_A T[*]\}$
**return** $S_{a_{\max}}$

**Subroutine 3** MarriageRep$(\Delta, T)$

select an lhs marriage $(X_1, X_2)$ of $\Delta$
**for all** $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2} T[*]$ **do**
  $S_{\mathbf{a}_1,\mathbf{a}_2} :=$ OptSRepair$(\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T, \Delta - X_1X_2)$
  $w(\mathbf{a}_1, \mathbf{a}_2) := w_T(S_{\mathbf{a}_1,\mathbf{a}_2})$
$V_i := \pi_{X_i} T[*]$ for $i = 1, 2$
$E := \{(\mathbf{a}_1, \mathbf{a}_2) \mid (\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2} T[*]\}$
$G :=$ weighted bipartite graph $(V_1, V_2, E, w)$
$E_{\max} :=$ a maximum matching of $G$
**return** $\cup_{(\mathbf{a}_1,\mathbf{a}_2) \in E_{\max}} S_{\mathbf{a}_1,\mathbf{a}_2}$

The algorithm handles four cases. The first is where $\Delta$ is trivial. Then, $T$ is itself an optimal S-repair. The second case is where $\Delta$ has a common lhs $A$. Then, the algorithm groups the tuples by $A$, finds an optimal S-repair for each group (via a recursive call to OptSRepair), this time by ignoring $A$ (i.e., removing $A$ from the FDs of $\Delta$), and returning the union of the optimal S-repairs. The precise description is in the subroutine CommonLHSRep (Subroutine 1). The third case is where $\Delta$ has a consensus FD $\emptyset \to A$. Similarly to the second case, the algorithm groups the tuples by $A$ and finds an optimal S-repair for each group. This time, however, the algorithm returns the optimal S-repair with the maximal weight. The precise description is in the subroutine ConsensusRep (Subroutine 2).

The fourth (last) case is the most involved. This is the case where $\Delta$ has an lhs marriage $(X_1, X_2)$. In this case the problem is reduced to finding a maximum weighted matching of a bipartite graph [26].

The graph, which we denote by $G = (V_1, V_2, E, w)$, consists of two disjoint node sets $V_1$ and $V_2$, an edge set $E$ that connects nodes from $V_1$ to nodes from $V_2$, and a weight function $w$ that assigns a weight $w(v_1, v_2)$ to each edge $(v_1, v_2)$. For $i = 1, 2$, the node set $V_i$ is the set of tuples in the projection of $T$ to $X_i$.[1] To determine the weight $w(v_1, v_2)$, we select from $T$ the subset $T_{v_1,v_2}$ that consists of the tuples that agree with $v_1$ and $v_2$ on $X_1$ and $X_2$, respectively. We then find an optimal S-repair for $T_{v_1,v_2}$, after we remove from $\Delta$ every attribute in either $X_1$ or $X_2$. Then, the weight $w(v_1, v_2)$ is the weight of this optimal S-repair. Next, we find a maximum matching $E_{\max}$ of $G$. Note that $E_{\max}$ is a subset of $E$ such that no node appears more than once. The returned result is then the disjoint union of the optimal S-repairs of $T_{v_1,v_2}$ over all $(v_1, v_2)$ in $E_{\max}$. The precise description is in the subroutine MarriageRep (Subroutine 3).

The following theorem states the correctness and efficiency of OptSRepair.

Theorem 3.2. *Let $\Delta$ and $T$ be a set of FDs and a table, respectively, over a relation schema $R(A_1, \ldots, A_k)$. If* OptSRepair$(\Delta, T)$ *succeeds, then it returns an optimal S-repair. Moreover,* OptSRepair$(\Delta, T)$ *terminates in polynomial time in $k$, $|\Delta|$, and $|T|$.*

What about the cases where OptSRepair$(\Delta, T)$ fails? We discuss it in the next section.

Next, we discuss the proof of Theorem 3.2. The proof of soundness is by induction on the number of simplifications that OptSRepair applies to $\Delta$. For each one of the three simplifications, we prove that if OptSRepair returns an optimal S-repair after the simplification is applied, then it also returns an optimal S-repair for the original set of FDs. We give a detailed proof just for the lhs-marriage simplification. The complete proof is in the archive version [28] and will be given in the full version of the paper.

Lemma 3.3. *Let $T$ be a table and $\Delta$ be a set of FDs that has an lhs marriage $(X_1, X_2)$. If* OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=a_1, X_2=a_2}T)$ *returns an optimal S-repair of $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$ w.r.t. $\Delta - X_1X_2$ for all $(\mathbf{a}_1, \mathbf{a}_2)$, then* MarriageRep$(\Delta, T)$ *returns an optimal S-repair of $T$ w.r.t. $\Delta$.*

Proof. Let $J$ be the result of MarriageRep$(\Delta, T)$. We first prove that $J$ is consistent. Assume, by way of contradiction, that $\mathbf{t}_1$ and $\mathbf{t}_2$ are two tuples in $J$ that jointly violate $\Delta$. We first observe that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ if and only if $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$, since $J$ is constructed via a matching of $G$ (lines 8-9). If $\mathbf{t}_1[X_1] \neq \mathbf{t}_2[X_1]$ and $\mathbf{t}_1[X_2] \neq \mathbf{t}_2[X_2]$, then the definition of an lhs marriage implies that $\mathbf{t}_1$ and $\mathbf{t}_2$ disagree on the left-hand side of every FD in $\Delta$, and hence satisfy $\Delta$. We conclude that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ and $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$, and therefore, $\mathbf{t}_1$ and $\mathbf{t}_2$ are both in $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$ for some $(\mathbf{a}_1, \mathbf{a}_2) \in \pi_{X_1X_2}T[*]$. Suppose that $\mathbf{t}_1$ and $\mathbf{t}_2$ violate the FD $Z \to W$ in $\Delta$. Since $\mathbf{t}_1$ and $\mathbf{t}_2$ agree on $X_1$ and $X_2$, they must violate $Z \setminus (X_1 \cup X_2) \to W \setminus (X_1 \cup X_2)$, which is in $\Delta - X_1X_2$. This contradicts the assumption that OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=a_1, X_2=a_2}T)$ returns an S-repair. We conclude that $J$ is $J$ is consistent, as claimed.

We complete the proof by showing that $J$ is optimal. Let $J'$ be a consistent subset of $T$. We need to prove that $w_T(J') \leq w_T(J)$. From the construction of $J$ it follows that $w_T(J) = w(E_{\max})$, where $w(E)$ denotes the sum of weights of a matching $E$ of $G$. So, it suffices to prove that $w_T(J') \leq w(E')$ for some matching $E'$ of $G$.

The definition of an lhs marriage implies that both $X_1 \to X_2$ and $X_2 \to X_1$ are entailed by $\Delta$. Hence, if $\mathbf{t}_1$ and $\mathbf{t}_2$ are tuples of $J'$, then it is again the case that $\mathbf{t}_1[X_1] = \mathbf{t}_2[X_1]$ if and only if $\mathbf{t}_1[X_2] = \mathbf{t}_2[X_2]$. We select as $E'$ the matching of $G$ that contains the edges $(\mathbf{a}_1, \mathbf{a}_2)$ whenever $\mathbf{t}[X_1] = \mathbf{a}_1$ and $\mathbf{t}[X_2] = \mathbf{a}_2$ for some tuple $\mathbf{t}$ of $J'$. As $J'$ is consistent, each of its subsets $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}J'$ is a consistent subset of $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$ w.r.t. $\Delta - X_1X_2$, since all the tuples in this subset agree on $X_1X_2$. Then, if $S_{\mathbf{a}_1,\mathbf{a}_2}$ is an optimal S-repair of $\sigma_{X_1=a_1, X_2=a_2}T$, then $w_T(\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}J') \leq w_T(S_{\mathbf{a}_1,\mathbf{a}_2})$. Moreover, $w_T(S_{\mathbf{a}_1,\mathbf{a}_2}) = w(\mathbf{a}_1, \mathbf{a}_2)$ due to the construction of $G$ and the assumption that OptSRepair$(\Delta - X_1X_2, \sigma_{X_1=a_1, X_2=a_2}T)$ returns an optimal S-repair. Thus, $w_T(J') \leq w(E')$ as claimed. □

Finally, we discuss the proof of the complexity claim of Theorem 3.2. The main observation here is that whenever the algorithm makes a recursive call, it is applied to disjoint sets of tuples of $T$. For example, the recurrence function for the subroutine MarriageRep looks as follows.

$$F(k, n) \;\leq\; P(k, n) + \sum_{\substack{(\mathbf{a}_1,\mathbf{a}_2)\in \\ \pi_{X_1X_2}T[*]}} F(k-1, n_{\mathbf{a}_1,\mathbf{a}_2})$$

where $k$ is the number of attributes that occur in $\Delta$, $n$ is the number of tuples in $T$, $P$ is a polynomial, and $n_{\mathbf{a}_1,\mathbf{a}_2}$ is the number of tuples in $\sigma_{X_1=\mathbf{a}_1, X_2=\mathbf{a}_2}T$. We again refer the reader to the archive version for the full proof [28].

*Approximation.* An easy observation is that the computation of an optimal subset is easily reducible to the *weighted vertex-cover* problem—given a graph $G$ where nodes are assigned nonnegative weights, find a vertex cover (i.e., a set $C$ of nodes that intersects with all edges) with a minimal sum of weights. Indeed, given a table $T$, we construct the graph $G$ that has $ids(T)$ as the set of nodes, and an edge between every $i$ and $j$ such that $T[i]$ and $T[j]$ contradict one or more FDs in $\Delta$. Given a vertex cover $C$ for $G$, we obtain a consistent subset $S$ by deleting from $T$ every tuple with an identifier in $C$. Clearly, this reduction is strict. As weighted vertex cover is 2-approximable in polynomial time [7], we conclude the same for optimal subset repairing.

Proposition 3.4. *For all FD sets $\Delta$, a 2-optimal S-repair can be computed in polynomial time.*

While Proposition 3.4 is straightforward, it is of practical importance as it limits the severity of the lower bounds we establish in the next section. Moreover, we will later show that the proposition has implications on the problem of approximating an optimal U-repair.

## 3.2 Dichotomy

The reader can observe that the success or failure of the algorithm OptSRepair$(\Delta, T)$ depends only on $\Delta$, and not on $T$. The algorithm OSRSucceeds$(\Delta)$, depicted as Algorithm 2, tests whether $\Delta$ is such that OptSRepair succeeds by simulating the cases and corresponding changes to $\Delta$. The next theorem shows that, under conventional complexity assumptions, OptSRepair covers *all* sets $\Delta$ such that an optimal S-repair can be found in polynomial time. Hence, we

[1] In principle, it may be the case that the same tuple occurs in both $V_1$ and $V_2$, since the tuple is in both projections. Nevertheless, we still treat the two occurrences of the tuple as distinct nodes, and so effectively assume that $V_1$ and $V_2$ are disjoint.

**Algorithm 2** OSRSucceeds($\Delta$)

```
1:  while Δ is nontrivial do
2:      remove trivial FDs from Δ
3:      if Δ has a common lhs A then
4:          Δ := Δ − A
5:      else if Δ has a consensus FD ∅ → A then
6:          Δ := Δ − A
7:      else if Δ has an lhs marriage (X1, X2) then
8:          Δ := Δ − X1X2
9:      else
10:         return false
11: return true
```

establish a dichotomy in the complexity of computing an optimal S-repair.

Theorem 3.5. *Let $\Delta$ be a set of FDs.*

- *If* OSRSucceeds($\Delta$) *returns true, then an optimal S-repair can be computed in polynomial time by executing* OptSRepair($\Delta, T$) *on the input $T$.*
- *If* OSRSucceeds($\Delta$) *returns false, then computing an optimal S-repair is APX-complete, and remains APX-complete on unweighted, duplicate-free tables.*

*Moreover, the execution of* OSRSucceeds($\Delta$) *terminates in polynomial time in $|\Delta|$.*

Recall that a problem in APX has a constant factor approximation and, under the assumption that P≠NP, an APX-hard problem cannot be approximated better than some constant factor (that may depend on the problem itself).

Example 3.6. We now illustrate the application of Theorem 3.5 to several FD sets. Consider first the FD set $\Delta$ of our running example. The execution of OSRSucceeds($\Delta$) transforms $\Delta$ as follows.

$$\begin{aligned} &\{\text{facility} \to \text{city},\ \text{facility room} \to \text{floor}\} \\ (\text{common lhs}) \Rightarrow &\{\emptyset \to \text{city},\ \text{room} \to \text{floor}\} \\ (\text{consensus}) \Rightarrow &\{\text{room} \to \text{floor}\} \\ (\text{common lhs}) \Rightarrow &\{\emptyset \to \text{floor}\} \\ (\text{consensus}) \Rightarrow &\{\} \end{aligned}$$

Hence, OSRSucceeds($\Delta$) is true, and hence, an optimal S-repair can be found in polynomial time.

Next, consider the FD set $\Delta_{A \leftrightarrow B \to C}$ from Example 3.1. The algorithm OSRSucceeds($\Delta_{A \leftrightarrow B \to C}$) executes as follows.

$$\begin{aligned} &\{A \to B, B \to A, B \to C\} \\ (\text{lhs marriage}) \Rightarrow &\{\emptyset \to C\} \\ (\text{consensus}) \Rightarrow &\{\} \end{aligned}$$

Hence, this is again an example of an FD set on the tractable side of the dichotomy.

As the last positive example we consider the FD set $\Delta_1$ of Example 3.1.

$$\begin{aligned} &\{\text{ssn} \to \text{first},\ \text{ssn} \to \text{last},\ \text{first last} \to \text{ssn},\ \text{ssn} \to \text{address}, \\ &\text{ssn office} \to \text{phone},\ \text{ssn office} \to \text{fax}\} \\ (\text{lhs marriage}) \Rightarrow &\{\emptyset \to \text{address},\ \text{office} \to \text{phone},\ \text{office} \to \text{fax}\} \\ (\text{consensus}) \Rightarrow &\{\text{office} \to \text{phone},\ \text{office} \to \text{fax}\} \\ (\text{common lhs}) \Rightarrow &\{\emptyset \to \text{phone},\ \emptyset \to \text{fax}\} \\ (\text{consensus}) \Rightarrow &\{\} \end{aligned}$$

On the other hand, for $\Delta = \{A \to B, B \to C\}$, none of the conditions of OSRSucceeds($\Delta$) is true, and therefore, the algorithm returns false. It thus follows from Theorem 3.5 that computing an optimal S-repair is APX-complete (even if all tuple weights are the same and there are no duplicate tuples). The same applies to $\Delta = \{A \to B, C \to D\}$. □

As another example, the following corollary of Theorem 3.5 generalizes the tractability of our running example to general chain FD sets.

Corollary 3.7. *If $\Delta$ is a chain FD set, then an optimal S-repair can be computed in polynomial time.*

Proof. The reader can easily verify that when $\Delta$ is a chain FD set, OSRSucceeds($\Delta$) will reduce it to emptiness by repeatedly removing consensus attributes and common-lhs, as done in our running example. □

## 3.3 Proof of Theorem 3.5

In this section, we discuss the proof of Theorem 3.5. (The full proof is in [28].) The positive side is a direct consequence of Theorem 3.2. For the negative side, membership in APX is due to Proposition 3.4. The proof of hardness is based on the concept of a *fact-wise reduction* [22], as previously done for proving dichotomies on sets of FDs [16, 22, 23, 27]. In our setup, a fact-wise reduction is defined as follows. Let $R$ and $R'$ be two relation schemas. A *tuple mapping* from $R$ to $R'$ is a function $\mu$ that maps tuples over $R$ to tuples over $R'$. We extend $\mu$ to map tables $T$ over $R$ to tables over $R'$ by defining $\mu(T)$ to be $\{\mu(t) \mid t \in T\}$. Let $\Delta$ and $\Delta'$ be sets of FDs over $R$ and $R'$, respectively. A *fact-wise reduction* from $(R, \Delta)$ to $(R', \Delta)$ is a tuple mapping $\Pi$ from $R$ to $R'$ with the following properties: *(a)* $\Pi$ is injective, that is, for all tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ over $R$, if $\Pi(\mathbf{t}_1) = \Pi(\mathbf{t}_2)$ then $\mathbf{t}_1 = \mathbf{t}_2$; *(b)* $\Pi$ preserves consistency and inconsistency; that is, $\Pi(T)$ satisfies $\Delta'$ if and only if $T$ satisfies $\Delta$; *and (c)* $\Pi$ is computable in polynomial time. The following lemma is straightforward.

Lemma 3.8. *Let $R$ and $R'$ be relation schemas and $\Delta$ and $\Delta'$ FD sets over $R$ and $R'$, respectively. If there is a fact-wise reduction from $(R, \Delta)$ to $(R', \Delta')$, then there is a strict reduction from the problem of computing an optimal S-repair under $R$ and $\Delta$ to that of computing an optimal S-repair under $R'$ and $\Delta'$.*

In the remainder of this section, we describe the way we use Lemma 3.8. Our proof consists of four steps.

(1) We first prove APX-hardness for each of the FD sets in Table 1 over $R(A, B, C)$. For $\Delta_{A \to B \to C}$ and $\Delta_{A \to C \leftarrow B}$ we adapt reductions by Gribkoff et al. [20] in a work that we discuss in Section 3.4. For $\Delta_{AB \to C \to B}$ we show a reduction from MAX-non-mixed-SAT [21]. Most intricate is the proof for

**Table 1: FD sets over $R(A, B, C)$ used in the proof of hardness of Theorem 3.5.**

| Name | FDs |
|---|---|
| $\Delta_{A\to B\to C}$ | $A \to B, B \to C$ |
| $\Delta_{A\to C\leftarrow B}$ | $A \to C, B \to C$ |
| $\Delta_{AB\to C\to B}$ | $AB \to C, C \to B$ |
| $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$ | $AB \to C, AC \to B, BC \to A$ |

$\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$, where we devise a nontrivial adaptation of a reduction by Amini et al. [3] to triangle packing in graphs of bounded degree.

(2) Next, we prove that whenever OSRSucceeds simplifies $\Delta$ into $\Delta'$, there is a fact-wise reduction from $(R, \Delta')$ to $(R, \Delta)$, where $R$ is the underlying relation schema.

(3) Then, we consider an FD set $\Delta$ that cannot be further simplified (that is, $\Delta$ does not have a common lhs, a consensus FD, or an lhs marriage). We show that $\Delta$ can be classified into one of five certain classes of FD sets (that we discuss next).

(4) Finally, we prove that for each FD set $\Delta$ in one of the five classes there exists a fact-wise reduction from one of the four schemas of Table 1.

The most challenging part of the proof is identifying the classes of FD sets in Step 3 in such a way that we are able to build the fact-wise reductions in Step 4. We first identify that if an FD set $\Delta$ cannot be simplified, then there are at least two distinct local minima $X_1 \to Y_1$ and $X_2 \to Y_2$ in $\Delta$. By a *local minimum* we mean an FD with a set-minimal lhs, that is, an FD $X \to Y$ such that no FD $Z \to W$ in $\Delta$ satisfies that $Z$ is a strict subset of $X$. We pick any two local minima from $\Delta$. Then, we divide the FD sets into five classes based on the relationships between $X_1, X_2, cl_\Delta(X_1) \setminus X_1$, which we denote by $\widehat{X_1}$, and $cl_\Delta(X_2) \setminus X_2$, which we denote by $\widehat{X_2}$. The classes are illustrated in Figure 2.

Each line in Figure 2 represents one of $X_1, X_2, \widehat{X_1}$ and $\widehat{X_2}$. If two lines do not overlap, then the corresponding two sets are assumed to be disjoint. For example, the sets $\widehat{X_1}$ and $\widehat{X_2}$ in class (1) are disjoint. Overlapping lines represent sets that have a nonempty intersection, an example being $\widehat{X_1}$ and $\widehat{X_2}$ in class (2). When two dashed lines overlap, we do not assume anything about their intersection. As an example, the sets $X_1$ and $X_2$ can have an empty or a nonempty intersection in each of the classes. Finally, if a line covers another line, then the set that corresponds to the first line contains that of the second line. For instance, the set $\widehat{X_2}$ in class (4) contains the set $X_1 \setminus X_2$, while in class (5) it holds that $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$. We remark that Figure 2 covers the important cases that we need to analyze, but it misses a few cases. (As previously said, details are in the extended version of this paper [28].)

Example 3.9. In what follows, we discuss the five classes of Figure 2. Specifically, we mention which FD set of Table 1 was used in the corresponding fact-wise reductions, and give an example of an FD set in the class.

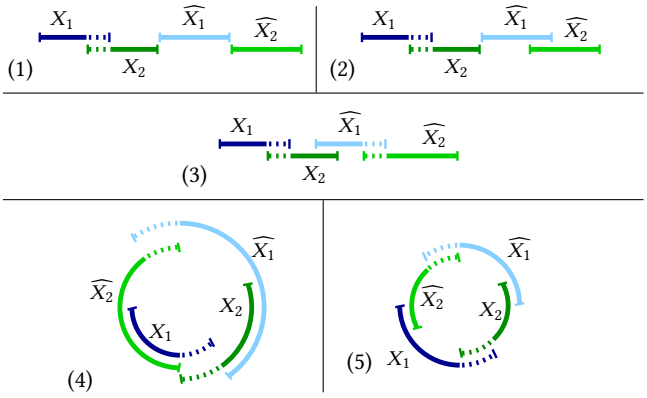


**Figure 2: Classes of FD sets that cannot be simplified.**

**Class 1.** We build fact-wise reductions from $\Delta_{A\to C\leftarrow B}$ to the FD sets in this class. An example of an FD set in this class is $\Delta_1 = \{A \to B, C \to D\}$. In this case $X_1 = \{A\}, X_2 = \{C\}, \widehat{X_1} = \{B\}$ and $\widehat{X_2} = \{D\}$. Thus, $\widehat{X_1} \cap X_2 = \emptyset$, $\widehat{X_2} \cap X_1 = \emptyset$ and $\widehat{X_1} \cap \widehat{X_2} = \emptyset$ and indeed the only overlapping lines in (1) are the dashed lines corresponding to $X_1$ and $X_2$.

**Class 2.** For this class, the fact-wise reductions are from $\Delta_{A\to B\to C}$. The FD set $\Delta_2 = \{A \to CD, B \to CE\}$, for example, belongs to this class. It holds that $X_1 = \{A\}, X_2 = \{B\}, \widehat{X_1} = \{C, D\}$ and $\widehat{X_2} = \{C, E\}$. Hence, $\widehat{X_1} \cap X_2 = \emptyset$ and $\widehat{X_2} \cap X_1 = \emptyset$, but $\widehat{X_1} \cap \widehat{X_2} \neq \emptyset$, and the difference from (1) is that the lines corresponding to $\widehat{X_1}$ and $\widehat{X_2}$ in (2) overlap.

**Class 3.** We again build fact-wise reductions from $\Delta_{A\to B\to C}$ to the FD sets in this class. As an example of an FD set in this class we use the set $\Delta_3 = \{A \to BC, B \to D\}$. Here, it holds that $X_1 = \{A\}$, $X_2 = \{B\}$, $\widehat{X_1} = \{B, C, D\}$ and $\widehat{X_2} = \{D\}$. Thus, $\widehat{X_1} \cap X_2 \neq \emptyset$, but $\widehat{X_2} \cap X_1 = \emptyset$. The difference from (2) is that now the lines corresponding to $X_2$ and $\widehat{X_1}$ overlap and we do not assume anything about the intersection between $\widehat{X_1}$ and $\widehat{X_2}$.

**Class 4.** Here, the fact-wise reductions are from $\Delta_{AB\leftrightarrow AC\leftrightarrow BC}$. An example of an FD set that belongs to this class is $\Delta_4 = \{A \to B, B \to C, C \to A\}$. In this case we have three local minima. We pick two of them: $A \to B$ and $B \to C$. Now, $X_1 = \{A\}, X_2 = \{B\}$, $\widehat{X_1} = \{B, C\}$ and $\widehat{X_2} = \{A, C\}$. Thus, $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$. The difference from (3) is that now the lines corresponding to $X_1$ and $\widehat{X_2}$ overlap. Moreover, the line corresponding to $\widehat{X_1}$ covers the entire line corresponding to $X_2 \setminus X_1$ and the line corresponding to $\widehat{X_2}$ covers the entire line corresponding to $X_1 \setminus X_2$. This means that we assume that $(X_1 \setminus X_2) \subseteq \widehat{X_2}$ and $(X_2 \setminus X_1) \subseteq \widehat{X_1}$.

**Class 5.** For FD sets in this class we build a fact-wise reduction from $\Delta_{AB\to C\to B}$. The FD set $\Delta_5 = \{AB \to C, C \to AD\}$ is an example of an FD set that belongs to this class. Here, $X_1 = \{A, B\}$, $X_2 = \{C\}$, $\widehat{X_1} = \{C, D\}$ and $\widehat{X_2} = \{A, D\}$, therefore $\widehat{X_1} \cap X_2 \neq \emptyset$ and $\widehat{X_2} \cap X_1 \neq \emptyset$. The difference from (4) is that now we assume that $(X_1 \setminus X_2) \not\subseteq \widehat{X_2}$.

Note that for each class we build infinitely many fact-wise reductions, one for each FD set in this class. To demonstrate a fact-wise

reduction, consider the FD set $\Delta_4 = \{A \to B, B \to C, C \to A\}$ given for Class 4. In the case of $\Delta_4$, the fact-wise reduction from $\Delta_{AB \leftrightarrow AC \leftrightarrow BC}$ is defined by

$$\Pi(a, b, c) \stackrel{\text{def}}{=} (\langle a, b\rangle, \langle b, c\rangle, \langle c, a\rangle)\,.$$

This reduction is a special case of a more involved fact-wise reduction that we defined for Class 4. □

### 3.4 Most Probable Database

In this section, we draw a connection to the *Most Probable Database problem* (MPD) [20]. A table in our setting can be viewed as a relation of a *tuple-independent database* [14] if each weight is in the interval $[0, 1]$. In that case, we view the weight as the probability of the corresponding tuple, and we call the table a *probabilistic table*. Such a table $T$ represents a probability space over the subsets of $T$, where a subset is selected by considering each tuple $T[i]$ independently and *selecting* it with the probability $w_T(i)$, or equivalently, deleting it with the probability $1 - w_T(i)$. Hence, the probability of a subset $S$, denoted $\Pr_T(S)$, is given by:

$$\Pr_T(S) \stackrel{\text{def}}{=} \left(\prod_{i \in ids(S)} w_T(i)\right) \times \left(\prod_{i \in ids(T) \setminus ids(S)} (1 - w_T(i))\right) \qquad (2)$$

Given a constraint $\varphi$ over the schema of $T$, MPD for $\varphi$ is the problem of computing a subset $S$ that satisfies $\varphi$, and has the maximal probability among all such subsets. Here, we consider the case where $\varphi$ is a set $\Delta$ of FDs. Hence, MPD for $\Delta$ is the problem of computing

$$\operatorname*{argmax}_{S \subseteq T,\, S \models \Delta} \Pr_T(S)\,.$$

Gribkoff, Van den Broeck, and Suciu [20] proved the following dichotomy for *unary* FDs, which are FDs of the form $A \to X$ having a single attribute on their lhs.

Theorem 3.10. [20] *Let $\Delta$ be a set of unary FDs over a relational schema. MPD for $\Delta$ is either solvable in polynomial time or NP-hard.*

The question of whether such a dichotomy holds for *general* (not necessarily *unary*) FDs has been left open. The following corollary of Theorem 3.5 fully resolves this question.

Theorem 3.11. *Let $\Delta$ be a set of FDs over a relational schema. If* OSRSucceeds($\Delta$) *is true, then MPD for $\Delta$ is solvable in polynomial time; otherwise, it is NP-hard.*

Proof. We first show a reduction from MPD to the problem of computing an optimal S-repair. Let $T$ be an input for MPD. By a *certain tuple* we refer to a tuple identifier $i \in ids(T)$ such that $w_T(i) = 1$. We assume that the set of certain tuples satisfies $\Delta$ collectively, since otherwise the probability of any consistent subset is zero (and we can select, e.g., the empty subset as a most likely solution). We can then replace each probability 1 with a probability that is smaller than, yet close enough to 1, so that every consistent subset that excludes a certain fact is less likely than any subset that includes all certain facts. In addition, as observed by Gribkoff et al. [20], tuples with probability at most 0.5 can be eliminated, since we can always remove them from any (consistent) subset without reducing the probability. Hence, we assume that $0.5 < w_T(i) < 1$ for all $i \in ids(T)$. From (2) we conclude the following.

$$\Pr_T(S) = \left(\prod_{i \in ids(S)} \frac{w_T(i)}{1 - w_T(i)}\right) \times \left(\prod_{i \in ids(T)} (1 - w_T(i))\right) \propto \left(\prod_{i \in ids(S)} \frac{w_T(i)}{1 - w_T(i)}\right)$$

The reason for the proportionality ($\propto$) is that all consistent subsets share the same right factor of the first product. Hence, we construct a table $T'$ that is the same as $T$, except that $w_{T'}(i) = \log(w_T(i)/(1 - w_T(i)))$ for all $i \in ids(T')$, and then a most likely database of $T$ is the same[2] as an optimal S-repair of $T'$.

For the "otherwise" part we show a reduction from the problem of computing an optimal S-repair of an *unweighted* table to MPD. The reduction is straightforward: given $T$, we set the weight $w_T(i)$ of each tuple to 0.9 (or any fixed number greater than 0.5). From (2) it follows that a consistent subset is most probable if and only if it has a maximal number of tuples. □

Comment 3.12. When considering unary FDs, there is a disagreement between our tractability condition (Algorithm 2) and that of Gribkoff et al. [20]. In particular, the FD set $\Delta_{A \leftrightarrow B \to C}$ defined in (1) is classified as polynomial time in our dichotomy while NP-hard by Gribkoff et al. [20]. This is due to a gap in their proof of hardness.[3] □

## 4 COMPUTING AN OPTIMAL U-REPAIR

In this section, we focus on the problem of computing an optimal U-repair and an approximation thereof. We give general results that assist in the complexity analysis of the problem, compare it to the problem of finding an optimal S-repair (discussed in the previous section), and identify sufficient conditions for efficient reductions between the two problems. Unlike S-repairs, the existence of a dichotomy for optimal U-repairs remains an open problem.

### 4.1 Reductions between FD Sets

For a set $\Delta$ of FDs, we use $attr(\Delta)$ to denote the set of attributes that appear in $\Delta$ (i.e., the union of lhs and rhs over all the FDs in $\Delta$). Two FD sets $\Delta_1$ and $\Delta_2$ (over the same schema) are *attribute disjoint* if $attr(\Delta_1)$ and $attr(\Delta_2)$ are disjoint. For example, $\{A \to BC, C \to D\}$ and $\{E \to FG\}$ are attribute disjoint. The following theorem implies that to determine the complexity of the union of two attribute-disjoint FD sets, it suffices to look at each set separately.

Theorem 4.1. *Suppose that $\Delta = \Delta_1 \cup \Delta_2$ where $\Delta_1$ and $\Delta_2$ are attribute disjoint. The following are equivalent for all $\alpha \geq 1$.*

*(1) An $\alpha$-optimal U-repair can be computed in polynomial time under $\Delta$.*
*(2) An $\alpha$-optimal U-repair can be computed in polynomial time under* each of $\Delta_1$ *and* $\Delta_2$.

Proof. *(Sketch)* The proof uses the following observation. If $U$, $U_1$ and $U_2$ are optimal U-repairs of $T$ w.r.t. $\Delta$, $\Delta_1$ and $\Delta_2$, respectively,

[2] We do not need to make an assumption of infinite precision to work with logarithms, since the algorithms we use for computing an optimal S-repair can replace addition and subtraction with multiplication and division, respectively.
[3] This has been established in a private communication with the authors of [20].

then $dist_{\mathsf{upd}}(U,T) = dist_{\mathsf{upd}}(U_1,T) + dist_{\mathsf{upd}}(U_2,T)$. This is true, since an optimal U-repair of a table $T$ w.r.t. $\Delta'$ keeps intact values of attributes outside of $\Delta'$.

To prove (1) $\rightarrow$ (2), we reduce the computation of a U-repair w.r.t. $\Delta_1$ to the computation of a U-repair w.r.t. $\Delta$, as follows. (The proof for $\Delta_2$ is symmetric.) Given $T$, we generate from $T$ a new table $T_1$ by keeping all values in the attributes of $\Delta_1$ intact, and changing the rest of the values to the same constant (hence, all dependencies of $\Delta_2$ are satisfied). We then find an $\alpha$-optimal U-repair $U$ of $T_1$ w.r.t. $\Delta$. Finally, we obtain an $\alpha$-optimal U-repair $U_1$ of $T$ w.r.t. $\Delta_1$ by using the values from $U$ for all the attributes of $\Delta_1$, and using the original values of $T$ for the remaining attributes.

For (2) $\rightarrow$ (1), we wish to compute an $\alpha$-optimal U-repair w.r.t. $\Delta$ for a given table $T$, assuming that an $\alpha$-optimal U-repair of $T$ can be computed in polynomial time w.r.t. both $\Delta_1$ and $\Delta_2$. We first construct $\alpha$-optimal U-repairs $U_1$ and $U_2$ of $T$ w.r.t. $\Delta_1$ and $\Delta_2$, respectively. Then, we obtain an $\alpha$-optimal U-repair $U$ w.r.t. $\Delta$ by using the values from $U_1$ for all the attributes in $attr(\Delta_1)$, the values from $U_2$ for all the attributes in $attr(\Delta_2)$, and the values from $T$ for the rest of the attributes. □

Example 4.2. Consider the following set of FDs.

$$\Delta \stackrel{\text{def}}{=} \{\mathsf{item} \rightarrow \mathsf{cost}\,,\ \mathsf{buyer} \rightarrow \mathsf{address}\}$$

We will later show that if $\Delta$ consists of a single FD, then an optimal U-repair can be computed in polynomial time. Hence, we can compute an optimal U-repair under $\Delta_1 = \{\mathsf{item} \rightarrow \mathsf{cost}\}$ and under $\Delta_2 = \{\mathsf{buyer} \rightarrow \mathsf{phone}\}$. Then, Theorem 4.1 implies that an optimal U-repair can be computed in polynomial time under $\Delta$ as well.

Now consider the following set of FDs.

$$\Delta' \stackrel{\text{def}}{=} \{\mathsf{item} \rightarrow \mathsf{cost}\,,\ \mathsf{buyer} \rightarrow \mathsf{address}\,,\ \mathsf{address} \rightarrow \mathsf{state}\}$$

Kolahi and Lakshmanan [24] proved that it is NP-hard to compute an optimal U-repair for $\{A \rightarrow B, B \rightarrow C\}$, by reduction from the problem of finding a minimum vertex cover of a graph $G$. Their reduction is, in fact, a PTAS reduction if we use vertex cover in a graph of a bounded degree [2]. Hence, computing an optimal U-repair is APX-hard for this set of FDs. Theorem 4.1 then implies that it is also APX-hard for $\Delta'$. □

Next, we discuss the problem in the presence of consensus FDs. The following theorem states that such FDs do not change the complexity of the problem. Recall that, for a set $\Delta$ of FDs and a set $X$ of attributes, the set $\Delta - X$ denotes the set of FDs that is obtained from $\Delta$ by removing each attribute of $X$ from the lhs and rhs of every FD. Also recall that $cl_\Delta(\emptyset)$ is the set of all consensus attributes.

Theorem 4.3. *Let $\Delta$ be a set of FDs. There is a strict reduction from computing an optimal U-repair for $\Delta$ to computing an optimal U-repair for $\Delta - cl_\Delta(\emptyset)$, and vice versa.*

The proof uses Theorem 4.1 and a special treatment of the case where all FDs are consensus. As an example of applying Theorem 4.3, if $\Delta$ consists of *only* consensus FDs, then an optimal U-repair can be computed in polynomial time, since $\Delta - cl_\Delta(\emptyset)$ is empty. As another example, if $\Delta$ is the set $\{\emptyset \rightarrow D, AD \rightarrow B, B \rightarrow CD\}$ then $\Delta - cl_\Delta(\emptyset) = \{A \rightarrow B, B \rightarrow C\}$ and, according to Theorem 4.3, computing an optimal U-repair is APX-hard, since this problem is hard for $\{A \rightarrow B, B \rightarrow C\}$ due to Kolahi and Lakshmanan [24], as explained in Example 4.2.

## 4.2 Reductions to/from Subset Repairing

In this section we establish several results that enable us to infer complexity results for the problem of computing an optimal U-repair from that of computing an optimal S-repair via polynomial-time reductions. We first need a notation.

Let $\Delta$ be a set of FDs. An *lhs cover* of $\Delta$ is a set $C$ of attributes that hits every nonempty lhs, that is, $X \cap C \neq \emptyset$ for every $X \rightarrow Y$ in $\Delta$, such that $X \neq \emptyset$. We denote the minimum cardinality of an lhs cover of $\Delta$ by $mlc(\Delta)$. For instance, if $\Delta$ is nonempty and has a common lhs (e.g., Figure 1), then $mlc(\Delta) = 1$.

The results of this section are based on the following proposition, which shows that we can transform a consistent update into a consistent subset (with no extra cost) and, in the absence of consensus FDs, a consistent subset into a consistent update (with some extra cost). We give the proof here, as it shows the actual constructions.

Proposition 4.4. *Let $\Delta$ be a set of FDs and $T$ a table. The following can be done in polynomial time.*

*(1) Given a consistent update $U$, construct a consistent subset $S$ such that $dist_{\mathsf{sub}}(S,T) \leq dist_{\mathsf{upd}}(U,T)$.*
*(2) Given a consistent subset $S$, and assuming that $\Delta$ is consensus free, construct a consistent update $U$ such that $dist_{\mathsf{upd}}(U,T) \leq mlc(\Delta) \cdot dist_{\mathsf{sub}}(S,T)$.*

Proof. We construct $S$ from $U$ by excluding any $i \in ids(T)$ such that $T[i]$ has at least one attribute updated in $U$ (i.e., $H(T[i], U[i]) \geq 1$). We construct $U$ from $S$ as follows. Let $C$ be an lhs cover of minimum cardinality $mlc(\Delta)$. The tuple of each $i \in ids(S)$ is left intact, and for $i \in ids(T) \setminus ids(S)$ we update the value of $T[i].A$ for each attribute $A \in C$ to a fresh constant from our infinite domain Val. Since $C$ is an lhs cover and there are no consensus FDs, for all $X \rightarrow Y$ in $\Delta$ it holds that two distinct tuples in $U$ that agree on $X$ must correspond to intact tuples; hence, $U$ is consistent (as $S$ is consistent). □

As we discuss later in Section 4.4, Proposition 4.4, combined with Proposition 3.4, reestablishes the result of Kolahi and Lakshmanan [24], stating that computing a U-repair is in APX. We also establish the following additional consequences of Proposition 4.4. The first is an immediate corollary (that we refer to later on) about the relationship between optimal repairs.

Corollary 4.5. *Let $\Delta$ be a set of FDs, $T$ a table, $S^*$ an optimal S-repair of $T$, and $U^*$ an optimal U-repair of $T$. Then, $dist_{\mathsf{sub}}(S^*,T) \leq dist_{\mathsf{upd}}(U^*,T)$. Moreover, if $\Delta$ is consensus free, then $dist_{\mathsf{upd}}(U^*,T) \leq mlc(\Delta) \cdot dist_{\mathsf{sub}}(S^*,T)$.*

The second consequence relates to FD sets $\Delta$ with a common lhs, that is, $mlc(\Delta) = 1$.

Corollary 4.6. *Let $\Delta$ be an FD set with a common lhs. There is a strict reduction from the problem of computing an optimal S-repair to that of computing an optimal U-repair, and vice versa.*

For example, if $\Delta$ consists of a single FD, then an optimal U-repair can be computed in polynomial time. Additional examples follow.

Example 4.7. To illustrate the use of Corollary 4.6, consider the FD set $\Delta$ of our running example (Figure 1). Since $\Delta$ has a common lhs, and we have established in Example 3.6 that an optimal S-repair for $\Delta$ can be found in polynomial time (i.e., $\Delta$ passes the test of OSRSucceeds), we get that an optimal U-repair can also be computed in polynomial time for $\Delta$.

As another illustration, consider the following FD set.

$$\Delta_1 \stackrel{\text{def}}{=} \{\text{id country} \rightarrow \text{passport}\,,\ \text{id passport} \rightarrow \text{country}\}$$

Again, $\Delta_1$ has a common lhs and $\Delta_1$ passes the test of OSRSucceeds (by applying common lhs followed by an lhs marriage), and therefore, Theorem 3.5 implies that an optimal U-repair can be found in polynomial time.

Finally, consider the following set of FDs.

$$\Delta_2 \stackrel{\text{def}}{=} \{\text{state city} \rightarrow \text{zip}\,,\ \text{state zip} \rightarrow \text{country}\}$$

The reader can verify that $\Delta_2$ fails OSRSucceeds, and therefore, from Theorem 3.5 we conclude that computing an optimal U-repair is APX-complete. □

By combining Theorem 4.3, Corollary 4.6, and Corollary 3.7, we conclude the following.

Corollary 4.8. *If $\Delta$ is a chain FD set, then an optimal U-repair can be computed in polynomial time.*

Proof. If $\Delta$ is a chain FD set, then so is $\Delta - cl_\Delta(\emptyset)$. Theorem 4.3 states that computing an optimal U-repair has the same complexity under the two FD sets. Moreover, if $\Delta - cl_\Delta(\emptyset)$ is nonempty, then it has at least one common lhs. From Corollary 4.6 we conclude that the problem then strictly reduces to computing an S-repair, which, by Corollary 3.7, can be done in polynomial time. □

Hence, for chain FD sets, an optimal repair can be computed for both subset and update variants.

## 4.3 Incomparability to S-Repairs

Corollaries 4.6 and 4.8 state cases where computing an optimal S-repair has the same complexity as computing an optimal U-repair. A basic case (among others) that the corollaries do not cover is in the following proposition, where again both variants have the same (polynomial-time) complexity.

Proposition 4.9. *Under $\Delta = \{A \rightarrow B, B \rightarrow A\}$, an optimal U-repair can be computed in polynomial time.*

In the proof of Proposition 4.9 we show that, even though $mlc(\Delta)$ is 2, we have $dist_{\text{upd}}(U^*, T) = dist_{\text{sub}}(S^*, T)$ for all tables $T$ over $R$, optimal U-repairs $U^*$ and optimal S-repairs $S^*$. Since $\Delta$ passes the test of OSRSucceeds (by applying lhs marriage), from Theorem 3.5 an optimal S-repair can be computed in polynomial time, and therefore an optimal U-repair of $\{A \rightarrow B, B \rightarrow A\}$ can also be computed in polynomial time.

Do the two variants of optimal repairs feature the same complexity for *every* set of FDs? Next, we answer this question in a negative way.

We have already seen an example of an FD set $\Delta$ where an optimal U-repair can be computed in polynomial time, but finding an S-repair is APX-complete. Indeed, Example 4.2 shows that $\{A \rightarrow B, C \rightarrow D\}$ is a tractable case for optimal U-repairs; yet, it fails the test of OSRSucceeds, and is therefore hard for optimal S-repairs (Theorem 3.5).

Showing an example of the other direction is more involved. The FD set in this example is $\Delta_{A \leftrightarrow B \rightarrow C}$ from Example 3.1. In Example 3.6 we showed that $\Delta_{A \leftrightarrow B \rightarrow C}$ passes the test of OSRSucceeds, and therefore, an optimal S-repair can be computed in polynomial time. This is not the case for an optimal U-repair.

Theorem 4.10. *For the relation schema $R(A, B, C)$ and the FD set $\Delta_{A \leftrightarrow B \rightarrow C}$, computing an optimal U-repair is APX-complete, even on unweighted, duplicate-free tables.*

The proof of hardness in Theorem 4.10 is inspired by, yet different from, the reduction of Kolahi and Lakshmanan [24] for showing hardness for $\{A \rightarrow B, B \rightarrow C\}$. The reduction is from the problem of finding a minimum vertex cover of a graph $G(V, E)$. Every edge $\{u, v\} \in E$ gives rise to the tuples $(u, v, 0)$ and $(v, u, 0)$. In addition, each vertex $v \in V$ gives rise to the tuple $(v, v, 1)$. The proof shows that there is a consistent update of cost at most $2|E| + k$ (assuming a unit weight for each tuple) if and only if $G$ has a vertex cover of size at most $k$. To establish a PTAS reduction, we use the APX-hardness of vertex cover when $G$ is of bounded degree [2]. The challenging (and interesting) part of the proof is in showing that a consistent update of cost $2|E| + k$ can be transformed into a vertex cover of size $k$; this part is considerably more involved than the corresponding proof of Kolahi and Lakshmanan [24].

We conclude with the following corollary, stating the existence of FD sets where the two variants of the problem feature different complexities.

Corollary 4.11. *There exist FD sets $\Delta_1$ and $\Delta_2$ such that:*

*(1) Under $\Delta_1$ an optimal S-repair can be computed in polynomial time, but computing an optimal U-repair is APX-complete.*
*(2) Under $\Delta_2$ an optimal U-repair can be computed in polynomial time, but computing an optimal S-repair is APX-complete.*

## 4.4 Approximation

In this section, we discuss approximations for optimal U-repairs. We restrict the discussion to FD sets $\Delta$ that are nonempty and consensus free. Note that this is not a limiting assumption, since empty $\Delta$s are trivially tractable, and consensus FDs can be eliminated, without increasing the approximation ratio, due to Theorem 4.3.

The combination of Propositions 3.4 and 4.4 gives the following.

Theorem 4.12. *An $\alpha$-optimal U-repair can be computed in polynomial time for $\alpha = 2 \cdot mlc(\Delta)$.*

Observe that the approximation ratio can be further improved by applying Theorem 4.1: if $\Delta$ is the union of attribute-disjoint FD sets $\Delta_1$ and $\Delta_2$, then an $\alpha$-optimal U-repair can be computed (under $\Delta$) where $\alpha = 2 \cdot \max\{mlc(\Delta_1), mlc(\Delta_2)\}$.

Kolahi and Lakshmanan [24] gave a constant-factor approximation algorithm for U-repairs (assuming $\Delta$ is fixed). We first explain their ratio, and then compare it to ours.

Let $\Delta$ be a set of FDs and assume (without loss of generality) that the rhs of each FD consists of a single attribute. By $MFS(\Delta)$ we denote the maximum number of attributes in the lhs of any FD in $\Delta$. An *implicant* of an attribute $A$ is a set $X$ of attributes such that $X \rightarrow A$ is entailed by $\Delta$. A *core implicant* of $A$ is a set $C$ of attributes

that hits every implicant of $A$ (i.e., $X \cap C \neq \emptyset$ whenever $X \rightarrow A$ is in the closure of $\Delta$). A *minimum core implicant* of $A$ is a core implicant of $A$ with the smallest cardinality. By $MCI(\Delta)$ we denote the size of the largest minimum core implicant over all attributes $A$.

Theorem 4.13. [24] *An $\alpha$-optimal U-repair can be computed in polynomial time where $\alpha = (MCI(\Delta) + 2) \cdot (2MFS(\Delta) - 1)$.*

In both Theorems 4.12 and 4.13, the approximation ratios are constants under data complexity, but depend on $\Delta$. It is still unknown whether there is a constant $\alpha$ that applies to all FD sets $\Delta$. Yet, it is known that a constant-ratio approximation cannot be obtained in polynomial time under *combined complexity* (where $R$, $T$, and $\Delta$ are all given as input) [24].

Although the proof of Theorem 4.12 is much simpler than the non-trivial proof of Theorem 4.13 given in [24], it can be noted that the approximation ratios in these two theorems are not directly comparable. If $k$ is the number of attributes, then the worst-case approximation ratio in Theorem 4.13 is quadratic in $k$, while the worst-case approximation in Theorem 4.12 is linear in $k$ (precisely, linear in $\min(k, |\Delta|)$). Moreover, an easy observation is that the ratio between the two approximation ratios can be at most linear in $k$. In the remainder of this section, we illustrate the difference between the approximations with examples.

First, we show an infinite sequence of FD sets where the approximation ratio of Theorem 4.12 is $\Theta(k)$ and that of Theorem 4.13 is $\Theta(k^2)$. For a natural number $k \geq 1$, we define $\Delta_k$ as follows.

$$\Delta_k \stackrel{\text{def}}{=} \{A_0 \cdots A_k \rightarrow B_0, B_0 \rightarrow C, B_1 \rightarrow A_0, \ldots, B_k \rightarrow A_0\}$$

The approximation ratio for $\Delta_k$ given by Theorem 4.12 is $2(k+2)$. For the approximation ratio of Theorem 4.13, we have $MFS(\Delta_k) = k + 1$ (due to the FD $A_0 \cdots A_k \rightarrow B$) and $MCI(\Delta_k) = k$ (since the core implicant of $A_0$ is $\{B_1, \ldots, B_k\}$). Hence, the approximation ratio of Theorem 4.13 grows quadratically with $k$ (i.e., it is $\Theta(k^2)$).

On the other hand, bellow is a sequence of FD sets in which the approximation ratio of Theorem 4.12 grows linearly with $k$, while that of Theorem 4.13 is a constant.

$$\Delta'_k \stackrel{\text{def}}{=} \{A_0A_1 \rightarrow B_0, A_1A_2 \rightarrow B_1, \ldots, A_kA_{k+1} \rightarrow B_k\}$$

Here, it holds that $mlc(\Delta'_k) = \lceil (k+1)/2 \rceil$, that $MFS(\Delta'_k) = 2$, and that $MCI(\Delta'_k) = 1$. Therefore, the approximation ratio of Theorem 4.12 is $\Theta(k)$ while that of Theorem 4.13 is constant.

The following theorem shows that computing an optimal U-repair for both $\Delta_k$ and $\Delta'_k$ is a hard problem, thus an approximation is, indeed, needed.

Theorem 4.14. *Let $k \geq 1$ be fixed. Computing an optimal U-repair is APX-complete for:*

*(1) $R(A_0, \ldots, A_k, B_0, \ldots, B_k, C)$ and $\Delta_k$;*
*(2) $R(A_0, \ldots, A_{k+1}, B_0, \ldots, B_k)$ and $\Delta'_k$.*

The proof for $\Delta_k$ is by a reduction from computing an optimal U-repair under $\{A \rightarrow B, B \rightarrow C\}$ (see Example 4.2). For $\Delta'_k$, we first show that the problem is APX-hard for $k = 1$. In this case, the FD set contains two FDs $A_0A_1 \rightarrow B_0$ and $A_1A_2 \rightarrow B_1$, thus $A_1$ is a common lhs, and Corollary 4.6, combined with the fact that computing an optimal S-repair under $\{A \rightarrow B, C \rightarrow D\}$ is APX-hard (Theorem 3.5), imply that computing an optimal U-repair is APX-hard as well. Then, we construct a reduction from computing an optimal U-repair under $\Delta'_k$ for $k = 1$ to computing an optimal U-repair under $\Delta'_k$ for $k > 1$.

Clearly, one can take the benefit of the approximations of both Theorems 4.12 and 4.13 by computing U-repairs by both algorithms and selecting the one with the smaller cost. As we showed, this combined approximation outperforms each of its two components.

## 5 DISCUSSION AND FUTURE WORK

We investigated the complexity of computing an optimal S-repair and an optimal U-repair. For the former, we established a dichotomy over all sets of FDs (and schemas). For the latter, we developed general techniques for complexity analysis, showed concrete complexity results, and explored the connection to the complexity of S-repairs. We presented approximation results and, in the case of U-repairs, compared to the approximation of Kolahi and Lakshmanan [24]. In the case of S-repairs, we drew a direct connection to probabilistic database repairs, and completed a dichotomy by Gribkoff et al. [20] to the entire space of FDs. Quite a few directions are left for future investigation, and we conclude with a discussion of some of these.

As our results are restricted to FDs, an obvious important direction is to extend our study to other types of integrity constraints, such as denial constraints [18], conditional FDs [10], referential constraints [15], and tuple-generating dependencies [8]. Moreover, the repair operations we considered are either exclusively tuple deletions or exclusively value updates. Hence, another clear direction is to allow mixtures of deletions, insertions and updates, where the cost depends on the operation type, the involved tuple, and the involved attribute (in the case of updates).

Our understanding of the complexity of computing an optimal U-repair is considerably more restricted than that of an optimal S-repair. We would like to complete our complexity analysis for optimal U-repairs into a full dichotomy. More fundamentally, we would like to incorporate restrictions on the allowed value updates. Our results are heavily based on the ability to update *any cell* with *any value* from an infinite domain. A natural restriction on the update repairs is to allow revising only certain attributes, possibly using a finite (small) space of possible new values. It is not clear how to incorporate such a restriction in our results and proof techniques.

In the case of S-repairs, we are interested in incorporating *preferences*, as in the framework of *prioritized repairing* by Staworko et al. [31]. There, priorities among tuples allow to eliminate subset repairs that are inferior to others (where "inferior" has several possible interpretations). It may be the case that priorities are rich enough to clean the database unambiguously [23]. A relevant question is, then, what is the minimal number of tuples that we need to delete in order to have an unambiguous repair? Alternatively, how many preferences are needed for this cause?

## Acknowledgments

The work of Benny Kimelfeld and Ester Livshits was supported by the Israel Science Foundation (ISF) Grant 1295/15. The work of Ester Livshits was also supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel Cyber Bureau. The work of Sudeepa Roy was supported by NSF awards IIS-1552538 and IIS-1703431, and NIH award 1R01EB025021-01.

## REFERENCES

[1] Foto N. Afrati and Phokion G. Kolaitis. 2009. Repair checking in inconsistent databases: algorithms and complexity. In *ICDT*. ACM, 31–41.
[2] Paola Alimonti and Viggo Kann. 2000. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.* 237, 1-2 (2000), 123–134.
[3] Omid Amini, Stéphane Pérennes, and Ignasi Sau. 2009. Hardness and approximation of traffic grooming. *Theor. Comput. Sci.* 410, 38-40 (2009), 3751–3760.
[4] Periklis Andritsos, Ariel Fuxman, and Renée J. Miller. 2006. Clean Answers over Dirty Databases: A Probabilistic Approach. In *ICDE*. IEEE Computer Society, 30.
[5] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *PODS*. 68–79.
[6] Ahmad Assadi, Tova Milo, and Slava Novgorodov. 2017. DANCE: Data Cleaning with Constraints and Experts. In *ICDE*. IEEE Computer Society, 1409–1410.
[7] Reuven Bar-Yehuda and Shimon Even. 1981. A Linear-Time Approximation Algorithm for the Weighted Vertex Cover Problem. *J. Algorithms* 2, 2 (1981), 198–203.
[8] Catriel Beeri and Moshe Y. Vardi. 1984. Formal Systems for Tuple and Equality Generating Dependencies. *SIAM J. Comput.* 13, 1 (1984), 76–98.
[9] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang-Chiew Tan. 2015. QOCO: A Query Oriented Data Cleaning System with Oracles. *PVLDB* 8, 12 (2015), 1900–1903.
[10] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional Functional Dependencies for Data Cleaning. In *ICDE*. IEEE, 746–755.
[11] Marco A. Casanova, Ronald Fagin, and Christos H. Papadimitriou. 1984. Inclusion Dependencies and Their Interaction with Functional Dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 29–59.
[12] Jan Chomicki and Jerzy Marcinkowski. 2005. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput.* 197, 1-2 (2005), 90–121.
[13] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: a commodity data cleaning system. In *SIGMOD*. ACM, 541–552.
[14] Nilesh N. Dalvi and Dan Suciu. 2004. Efficient Query Evaluation on Probabilistic Databases. In *VLDB*. Morgan Kaufmann, 864–875.
[15] C. J. Date. 1981. Referential Integrity. In *VLDB*. VLDB Endowment, 2–12.
[16] Ronald Fagin, Benny Kimelfeld, and Phokion G. Kolaitis. 2015. Dichotomies in the Complexity of Preferred Repairs. In *PODS*. ACM, 3–15.
[17] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management*. Morgan & Claypool Publishers.
[18] Terry Gaasterland, Parke Godfrey, and Jack Minker. 1992. An Overview of Cooperative Answering. *J. Intell. Inf. Syst.* 1, 2 (1992), 123–157.
[19] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC Data-Cleaning Framework. *PVLDB* 6, 9 (2013), 625–636.
[20] Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. 2014. The Most Probable Database Problem. In *BUDA*.
[21] Johan Håstad. 2001. Some optimal inapproximability results. *J. ACM* 48, 4 (2001), 798–859.
[22] Benny Kimelfeld. 2012. A dichotomy in the complexity of deletion propagation with functional dependencies. In *PODS*. 191–202.
[23] Benny Kimelfeld, Ester Livshits, and Liat Peterfreund. 2017. Detecting Ambiguity in Prioritized Database Repairing. In *ICDT*. 17:1–17:20.
[24] Solmaz Kolahi and Laks V. S. Lakshmanan. 2009. On approximating optimum repairs for functional dependency violations. In *ICDT*, Vol. 361. ACM, 53–62.
[25] Mark W. Krentel. 1988. The Complexity of Optimization Problems. *J. Comput. Syst. Sci.* 36, 3 (1988), 490–509.
[26] H. W. Kuhn and Bryn Yaw. 1955. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart* (1955), 83–97.
[27] Ester Livshits and Benny Kimelfeld. 2017. Counting and Enumerating (Preferred) Database Repairs. In *PODS*. 289–301.
[28] Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. 2017. Computing Optimal Repairs for Functional Dependencies. *CoRR* abs/1712.07705 (2017). arXiv:1712.07705 http://arxiv.org/abs/1712.07705
[29] Andrei Lopatenko and Leopoldo E. Bertossi. 2007. Complexity of Consistent Query Answering in Databases Under Cardinality-Based and Incremental Repair Semantics. In *ICDT*. 179–193.
[30] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201.
[31] Slawek Staworko, Jan Chomicki, and Jerzy Marcinkowski. 2012. Prioritized repairing and consistent query answering in relational databases. *Ann. Math. Artif. Intell.* 64, 2-3 (2012), 209–246.
[32] Dan Suciu, Dan Olteanu, R. Christopher, and Christoph Koch. 2011. *Probabilistic Databases* (1st ed.). Morgan & Claypool Publishers.