Database Repairing with Soft Functional Dependencies

Nofar Carmeli
Technion - Israel Institute of Technology, Haifa, Israel

Martin Grohe
RWTH Aachen University, Germany

Benny Kimelfeld
Technion - Israel Institute of Technology, Haifa, Israel

Ester Livshits
Technion - Israel Institute of Technology, Haifa, Israel

Muhammad Tibi
Technion - Israel Institute of Technology, Haifa, Israel

Abstract

A common interpretation of soft constraints penalizes the database for every violation of every constraint, where the penalty is the cost (weight) of the constraint. A computational challenge is that of finding an optimal subset: a collection of database tuples that minimizes the total penalty when each tuple has a cost of being excluded. When the constraints are strict (i.e., have an infinite cost), this subset is a "cardinality repair" of an inconsistent database; in soft interpretations, this subset corresponds to a "most probable world" of a probabilistic database, a "most likely intention" of a probabilistic unclean database, and so on. Within the class of functional dependencies, the complexity of finding a cardinality repair is thoroughly understood. Yet, very little is known about the complexity of finding an optimal subset for the more general soft semantics. This paper makes a significant progress in this direction. In addition to general insights about the hardness and approximability of the problem, we present algorithms for two special cases: a single functional dependency, and a bipartite matching. The latter is the problem of finding an optimal "almost matching" of a bipartite graph where a penalty is paid for every lost edge and every violation of monogamy.

2012 ACM Subject Classification Theory of computation → Incomplete, inconsistent, and uncertain databases; Information systems → Data cleaning

Keywords and phrases Database inconsistency, database repairs, integrity constraints, soft constraints, functional dependencies

Digital Object Identifier 10.4230/LIPIcs.ICDT.2021.16

Funding This work was supported by the German Research Foundation (DFG) Project 412400621 (DIP program) and the Israel Science Foundation (ISF), Grant 768/19. The work of Nofar Carmeli was supported by the Google PhD Fellowship.

Acknowledgements The authors are very grateful to Alessio Conte and Peter Lindner for insightful discussions about the work described in this paper.

1 Introduction

Various challenges in data management are based on soft variants of database constraints (also referred to as *weak* or *approximate* constraints). In constraint discovery and mining, for instance, the goal is to find constraints, such as Functional Dependencies (FDs) [3, 8, 11] and beyond [2, 12, 16], that generally hold in the database but not necessarily in a perfect manner. There, the reason for the violations might be rare events (e.g., agreement on the zip code but

not the state) or noise (e.g., mistyping). Soft constraints also arise when reasoning about uncertain data [6, 9, 18, 19] – the database is viewed as a probabilistic space over possible worlds, and the violation of a weak constraint in a possible world is viewed as evidence that affects the world's probability.

Our investigation concerns the latter application of soft constraints. To be more precise, the semantics is that of a *parametric factor graph*: the probability of a possible world is the product of *factors* where every violation of the constraint contributes one factor; in turn, this factor is a weight that is assigned upfront to the constraint. This approach is highly inspired by successful concepts such as the Markov Logic Network (MLN) [17].[1] The computational problems are the typical ones of probabilistic modeling: marginal inference (compute the probability of a query answer) and maximum likelihood (find the most probable world) – the problem that we focus on here.

More specifically, we investigate the complexity of finding a most probable world in the case where the constraints are FDs. By taking the logarithms of the factors, this problem can be formally defined as follows. We are given a database D and a set Δ of FDs, where every tuple and every FD has a weight (which is a nonnegative number). We wish to obtain a cleaner subset E of D by deleting tuples. The cost of the subset E includes a penalty for every deleted tuple, and a penalty for every violation of (i.e., pair of tuples that jointly violate) an FD; the penalties are the weights of the tuple and the FD, respectively. The goal is to find a subset E with a minimal cost. In what follows, we refer to such E as an *optimal subset* and to the optimization problem of finding an optimal subset as *soft repairing*. The optimal subset corresponds to the "most likely intention" in the Probabilistic Unclean Database (PUD) framework of De Sa, Ilyas, Kimelfeld, Ré and Rekatsinas [18] in a restricted case that is studied in their work, and to the "most probable world" in the probabilistic database model of Sen, Deshpande and Getoor [19]. In the special case where the FDs are hard constraints (i.e., their weight is infinite or just too large to pay), an optimal subset is simply what is known as a "cardinality repair" [15] or, equivalently [14], a "most probable database" [6].

The computational challenge of soft repairing is that there are exponentially many candidate subsets. We investigate the data complexity of the problem, where the database schema and the FD set are fixed, and the input consists of the database D and all involved weights. Moreover, we assume that D consists of a single relation; this is done without loss of generality, since the problem boils down to soft repairing each relation independently (since an FD does not involve more than one relation).

The complexity of the problem is very well understood in the case of hard constraints (cardinality repairs). Gribkoff, Van den Broeck and Suciu [6] established complexity results for the case of unary FDs (having a single attribute on the left-hand side), and Livshits, Kimelfeld and Roy [14] completed the picture to a full (effective) dichotomy over all possible sets of FDs. For example, the problem is solvable in polynomial time for the FD sets $\{A \to B\}$, $\{A \to B, B \to A\}$ and $\{A \to B, B \to A, B \to C\}$, but is NP-hard for $\{A \to B, B \to C\}$. In contrast, very little is known about the more general case where the FDs are soft (and violations are allowed), where the problem seems to be fundamentally harder, both to solve and to reason about. Clearly, for every Δ where it is intractable to find a cardinality repair, the soft version is also intractable. But the other direction is false (under conventional complexity assumptions). For example, soft repairing is hard for $\Delta = \{A \to B, B \to A, B \to C\}$, for the

[1] More precisely, an MLN can be viewed as a database with weak constraints, where the set of facts includes all possible tuples over the (finite) domains of the attributes.

following reason. We can set the weights of $A \to B$ and $B \to C$ to be very high, making each of them a hard constraint in effect, and the weight of $B \to A$ very low, making it ignorable in effect. Hence, an optimal subset is a cardinality repair for $\{A \to B, B \to C\}$ that, as said above, is hard to compute.

So, which sets of FDs have a tractable soft repairing? The only polynomial-time algorithm we are aware of is that of De Sa et al. [18] for the special case of a single key constraint, that is, $\Delta = \{X \to Y\}$ where XY contain all of the schema attributes; they have left the more general case (that we study here) open. In this work, we make substantial progress in answering this question by presenting algorithms for two types of FD sets: *(a)* a single FD *and (b)* a matching constraint.

The first type generalizes the tractability of De Sa et al. [18] from a key constraint to an arbitrary FD (as long as it is the only FD in Δ). Like theirs, our algorithm employs dynamic programming, but in a more involved fashion. This is because their algorithm is based on the fact that in a key constraint $X \to Y$, any two tuples that agree on X are necessarily conflicting. We also show that our algorithm can be generalized to additional sets of FDs. For example, it turns out that the FD set $\{\text{name} \to \text{address}\,,\ \text{name address} \to \text{email}\}$ is tractable as well. (Note that the address attribute on the left-hand side of the second FD is not redundant, as in the ordinary semantics, since the FDs are treated as soft constraints.) In Section 4 we phrase the more general condition that this FD set satisfies.

The second type, matching constraints, refers to FD sets $\Delta = \{X \to Y, X' \to Y'\}$ over a schema with the attributes $A_1, \ldots, A_k$ where $X \cup Y = X' \cup Y' = X \cup X' = \{A_1, \ldots, A_k\}$ and there are no attributes other than $A_1, \ldots, A_k$. The simplest example is $\{A \to B, B \to A\}$ over the binary schema (A, B) that represents a bipartite graph, and the problem is that of finding the best "almost matching" of a bipartite graph where a penalty is paid for every lost edge and every violation of monogamy. A more involved example is $\{\text{fn ln} \to \text{addr}\,,\ \text{fn addr} \to \text{ln}\}$ over the schema $(\text{fn}, \text{ln}, \text{addr})$. Our algorithm is based on a reduction to the *Minimum Cost Maximum Flow* (MCMF) problem [4].

Whether our algorithms cover all of tractable cases remains an open problem for future investigation. (In the Conclusions we discuss the simplest FD sets where the question is left unsolved.) We do show, however, that there is a polynomial-time approximation algorithm with an approximation factor 3, that is, a subset where the penalty is at most three times the optimum.

The rest of the paper is organized as follows. We give the formal setup and the problem definition in Section 2. We then discuss the complexity of the general problem and its relationship to past results in Section 3. We describe our algorithm for soft repairing in Sections 4 and 5 for a single FD and a matching constraint, respectively, and conclude in Section 6.

2 Formal Setup

We begin with preliminary definitions and terminology that we use throughout the paper.

2.1 Databases, FDs and Repairs

A *relation schema* $R(A_1, \ldots, A_k)$ consists of a relation symbol R and a set $\{A_1, \ldots, A_k\}$ of attributes. A *database* D over R is a set of facts f of the form $R(c_1, \ldots, c_k)$, where each c_i is a *constant*. We denote by $f[A_i]$ the value that the fact f associates with attribute A_i (i.e., $f[A_i] = c_i$). Similarly, if $X = B_1 \cdots B_k$ is a sequence of attributes from $\{A_1, \ldots, A_k\}$, then

$f[X]$ is the tuple $(f[B_1], \ldots, f[B_k])$. We assume that every fact $f \in D$ is associated with a nonnegative weight, hereafter denoted w_f. (The weight of a fact is sometimes derived from a validity/existence probability [6, 19].)

A *Functional Dependency* (FD) over the relation schema $R(A_1, \ldots, A_k)$ is an expression φ of the form $X \to Y$ where $X, Y \subseteq \{A_1, \ldots, A_k\}$. A *violation* of an FD in a database D is a pair $\{f, g\}$ of tuples from D that agrees on the left-hand side (i.e., $f[X] = g[X]$) but disagrees on the right-hand side (i.e., $f[Y] \neq g[Y]$). An FD $X \to Y$ is *trivial* if $Y \subseteq X$. We denote by $\mathsf{vio}(D, \varphi)$ the set of all the violations of the FD φ in D. We say that D *satisfies* φ, denoted $D \models \varphi$, if it has no violations (i.e., $\mathsf{vio}(D, \varphi)$ is empty). The database D satisfies a set Δ of FDs, denoted by $D \models \Delta$, if D satisfies every FD in Δ; otherwise, D violates Δ (denoted $D \not\models \Delta$). We assume that every FD $\varphi \in \Delta$ has a nonnegative weight, that we denote by w_φ.

When there is no risk of ambiguity, we may omit the specification of the relation schema $R(A_1, \ldots, A_k)$ and simply assume that the involved databases and constraints are all over the same schema.

Let D be a database and let Δ be a set of FDs. A *repair* (*of D w.r.t. Δ*) is an inclusion-maximal consistent subset E; that is, $E \subseteq D$ and $E \models \Delta$, and moreover, $E' \not\models \Delta$ for every E' such that $E \subsetneq E' \subseteq D$. Note that the number of repairs can be exponential in the number of facts of D. A *cardinality repair* is a repair E of a maximal cardinality (i.e., $|E| \geq |E'|$ for every repair E').

2.2 Soft Constraints

We define the concept of *soft constraints* (or *weak constraints* or *weighted rules*) in the standard way of "penalizing" the database for every missing fact, on the one hand, and every violation, on the other hand. This is the concept adopted in past work such as the *parfactors* of De Sa et al. [18], the *soft keys* of Jha et al. [9], and the *PrDB* model of Sen et al. [19]. The concept can be viewed as a special case of the *Markov Logic Network* (MLN) [17].

Formally, let D be a database and Δ a set of FDs. The *cost* of a subset E of a database D is then defined as follows.

$$\mathsf{cost}(E \mid D) \stackrel{\text{def}}{=} \left(\sum_{f \in (D \setminus E)} w_f \right) + \left(\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)| \right) \tag{1}$$

As for the computational model, we assume that every weight is a rational number r/q that is represented using the numerator and the denominator, namely (r, q), where each of the two is an integer represented in the standard binary manner.

2.3 Problem Definition: Soft Repairing

The problem we study in this paper, referred to as *soft repairing*, is the optimization problem of finding a database subset with a minimal cost. Since we consider the data complexity of the problem, we associate with each relation schema and set of FDs a separate computational problem.

▶ **Problem 1** (Soft Repairing). *Let $R(A_1, \ldots, A_k)$ be a relation schema and Δ a set of FDs.* Soft repairing *(*for $R(A_1, \ldots, A_k)$ and Δ*) is the following optimization problem: Given a database D, find an optimal subset of D, that is, a subset E of D with a minimal* $\mathsf{cost}(E \mid D)$.

FLIGHTS

Flight	Airline	Date	Origin	Destination	Airplane	
UA123	United Airlines	01/01/2021	LA	NY	N652NW	3
UA123	United Airlines	01/01/2021	NY	UT	N652NW	2
UA123	Delta	01/01/2021	LA	NY	N652NW	1
DL456	Southwest	02/01/2021	NC	MA	N713DX	2
DL456	Southwest	03/01/2021	NJ	FL	N245DX	1
DL456	Delta	03/01/2021	CA	IL	N819US	4

(a) D.

FLIGHTS

Flight	Airline	Date	Origin	Destination	Airplane	
UA123	United Airlines	01/01/2021	NY	UT	N652NW	2
DL456	Southwest	02/01/2021	NC	MA	N713DX	2
DL456	Southwest	03/01/2021	NJ	FL	N245DX	1

(b) E_1.

FLIGHTS

Flight	Airline	Date	Origin	Destination	Airplane	
UA123	United Airlines	01/01/2021	LA	NY	N652NW	3
DL456	Delta	03/01/2021	CA	IL	N819US	4

(c) E_2.

FLIGHTS

Flight	Airline	Date	Origin	Destination	Airplane	
UA123	United Airlines	01/01/2021	LA	NY	N652NW	3
UA123	United Airlines	01/01/2021	NY	UT	N652NW	2
DL456	Delta	03/01/2021	CA	IL	N819US	4

(d) E_3.

Figure 1 For the relation FLIGHTS(Flight, Airline, Date, Origin, Destination, Airplane) and the FDs Flight $\rightarrow$ Airline (with $w_{\varphi_1} = 5$) and Flight Airline Date $\rightarrow$ Destination (with $w_{\varphi_2} = 1$), a database D, a cardinality repair E_1, a weighted cardinality repair E_2, and an optimal subset E_3.

Note that a cardinality repair is an optimal subset in the special case where the weight w_φ of every FD φ is ∞ (or just higher than the cost of deleting the entire database), and the weight w_f of every fact f is 1. Livshits et al. [14] studied the complexity of finding a *weighted cardinality repair*, which is the same as a cardinality repair but the weight w_f of every fact f can be arbitrary. Hence, both types of cardinality repairs are consistent (i.e., the constraints are strictly satisfied). In contrast, an optimal subset in the general case may violate one or more of the FDs. In the next section we recall the known complexity results for cardinality and weighted cardinality repairs.

▶ **Example 2.** Our running example is based on the database of Figure 1 over the relation schema FLIGHTS(Flight, Airline, Date, Origin, Destination, Airplane) that contains information about domestic flights in the United States. The weight of each tuple appears on the rightmost column. The FD set Δ consists of the following FDs:

- Flight $\rightarrow$ Airline: a flight is associated with a single airline.
- Flight Airline Date $\rightarrow$ Destination: a flight on a certain date has a single destination.

We assume that the weight of the first FD is 5, and the weight of the second FD is 1 (as the same flight number can be reused for different flights).

The database E_1 of Figure 1 is a cardinality repair of D as no repair of D can be obtained by removing less then three facts. However, E_1 is not a weighted cardinality repair, since its

Algorithm 1 Simplify(Δ).

Remove trivial FDs from Δ
if Δ is not empty **then**
  find a removable pair (X, Y) of attribute set
  $\Delta := \Delta - XY$
return Δ

cost is eight, while the cost of E_2 is six. The reader can easily verify that E_2 is a weighted cardinality repair of D. Finally, E_3 is not a repair of D in the traditional sense as it contains a violation of the second FD, but it is an optimal subset of D with $\mathsf{cost}(E_3 \mid D) = 5$. ⌟

3 Preliminary Complexity Analysis

We consider the data complexity of the problem of computing an optimal subset. We assume that the schema and the set of FDs are fixed, and the input consists of the database. Livshits et al. [14] studied the problems of finding a cardinality repair and a weighted cardinality repair, and established a dichotomy over the space of all the sets of functional dependencies. In particular, they introduced an algorithm that, given a set Δ of FDs, decides whether:

1. A weighted cardinality repair can be computed in polynomial time; *or*
2. Finding a (weighted) cardinality repair is APX-complete.[2]

No other possibility exists. The algorithm is a recursive procedure that attempts to simplify Δ at each iteration by finding a *removable* pair (X, Y) of attribute sets, and removing every attribute of X and Y from all the FDs in Δ (which we denote by $\Delta - XY$). We say that a pair (X, Y) of attribute sets is removable if it satisfies the following properties:

- $\mathrm{Closure}_\Delta(X) = \mathrm{Closure}_\Delta(Y)$,
- XY is nonempty,
- every FD in Δ contains either X or Y on the left-hand side.

Note that the sets X and Y may be the same, and then the condition states that every FD contains X on the left-hand side.

The simplification procedure for an FD set Δ is depicted here as Algorithm 1. If we are able to transform Δ to an empty set of FDs by repeatedly applying simplifications, then the algorithm returns true and finding an optimal consistent subset is solvable in polynomial time. Otherwise, the algorithm returns false and the problem is APX-complete. We state their result for later reference.

▶ **Theorem 3.** [14] *Let Δ be a set of FDs. If Δ can be emptied via* Simplify(Δ) *steps, then a weighted cardinality repair can be computed in polynomial time; otherwise, finding a cardinality repair is APX-complete.*

The hardness side of Theorem 3 immediately implies the hardness of the more general soft-repairing problem. Yet, the other direction (tractability generalizes) is not necessarily true. As discussed in the Introduction, if $\Delta = \{A \to B, B \to A, B \to C\}$, then Δ, as a set of hard constraints, is classified as tractable according to Algorithm 1; however, this is not the case for soft constraints. We can generalize this example by stating that if Δ contains

[2] Recall that APX is the class of NP optimization problems that admit constant-ratio approximations in polynomial time. Hardness in APX is via the so called "PTAS" reductions (cf. textbooks on approximation complexity, e.g., [5]).

a *subset* that is hard according to Theorem 3, then soft repairing is hard. (This does not hold when considering only hard constraints, as the example shows that there exists an easy Δ with a hard subset.) In the following sections, we are going to discuss tractable cases of FD sets. Before that, we will show that the problem becomes tractable if one settles for an approximation.

3.1 Approximation

The following theorem shows that soft repairing admits a constant-ratio *approximation*, for the constant three, in polynomial time. This means that there is a polynomial-time algorithm for finding a subset with a cost of at most three times the minimum.

▶ **Theorem 4.** *For all FD sets, soft repairing admits a 3-approximation in polynomial time.*

Proof. We reduce soft repairing to the problem of finding a minimum weighted set cover where every element belongs to 3 sets. A simple greedy algorithm finds a 3-approximation to this problem in linear time [7].

We set the elements to be $\{(\{f, g\}, \delta) \mid f, g \in D, \delta \in \Delta, f \text{ and } g \text{ contradict } \delta\}$. Each element $(\{f, g\}, \delta)$ belongs to three sets: f with weight w_f, g with weight w_g, and $(\{f, g\}, \delta)$ with weight w_δ. Each minimal solution to this set cover problem can be translated to a soft repair: the selected sets that correspond to tuples are removed in the repair. Indeed, a minimal set cover of such a construction has to resolve each conflict by either paying for the removal of at least one of the tuples or paying for the violation. ◀

In terms of formal complexity, Theorem 4 implies that the problem of soft repairing is in APX (for every set of FDs). From this, from Theorem 3 and from the discussion that follows Theorem 3, we conclude the following.

▶ **Corollary 5.** *Let Δ be a set of FDs. Soft repairing for Δ is in APX. Moreover, if any subset of Δ cannot be emptied via* Simplify(Δ) *steps, then soft repairing is APX-complete for Δ.*

4 Algorithm for a Single Functional Dependency

In this section, we consider the case of a single functional dependency, and present a polynomial-time algorithm for soft repairing. Hence, we establish the following result.

▶ **Theorem 6.** *In the case of a single FD, soft repairing can be solved in polynomial time.*

Next, we prove Theorem 6 by presenting an algorithm. Later, we also generalize the argument and result beyond a single FD (Theorem 7).

We assume that the single FD is $\varphi \stackrel{\text{def}}{=} X \to Y$ and that our input database is D. We split D into *blocks* and *subblocks*, as we explain next. The blocks of D are the maximal subsets of D that agree on the X values. Denote these blocks by $D_1, \ldots, D_m$. Note that there are no conflicts across blocks; hence, we can solve the problem separately for each block and then an optimal subset E is simply the union of optimal subsets E_i of the blocks D_i:

$$E = \bigcup_{i=1}^{m} E_i$$

The subblocks of a block D_i are the maximal subsets of D_i that agree on the Y values (in addition to the X values). We denote these subblocks by $D_{i,1}, \ldots, D_{i,q_i}$. Note that two facts from the same subblock are consistent, while two facts from different subblocks are conflicting.

From here we continue with dynamic programming. For a number $j \in \{0, \ldots, q_i\}$, where q_i is the number of subblocks of D_i, and a number $k \in \{0, \ldots, |D_{i,1} \cup \cdots \cup D_{i,j}|\}$ of facts, we define the following values that we are going to compute:

- $C[i, j, k]$ is the cost of an optimal subset of $D_{i,1} \cup \cdots \cup D_{i,j}$ (i.e., the union of the first j subblocks) with precisely k facts.
- $F[i, j, k]$ is a subset of $D_{i,1} \cup \cdots \cup D_{i,j}$ that realizes $C[i, j, k]$, that is,

$$|F[i, j, k]| = k \quad \wedge \quad \mathsf{cost}\left(F[i, j, k] \mid D_{i,1} \cup \cdots \cup D_{i,j}\right) = C[i, j, k]$$

(If multiple choices of $F[i, j, k]$ exist, we select an arbitrary one.) Once we compute the $F[i, q_i, k]$, we are done since it then suffices to return the best subset over all k:

$$E_i = F[i, q_i, k] \text{ for } k = \operatorname*{argmin}_k C[i, q_i, k]$$

It remains to compute $C[i, j, k]$ and $F[i, j, k]$. We will focus on the former, as the latter is obtained by straightforward bookkeeping. The key observation is that if we decide to delete t facts from $D_{i,j}$, then we always prefer to delete the t facts with the minimal weight. We use this observation as follows.

For a subblock $D_{i,j}$ and $t \in \{0, \ldots, |D_{i,j}|\}$, denote by $\mathsf{top}(t, D_{i,j})$ an arbitrary subset of $D_{i,j}$ with t facts of the highest weight. Hence, $\mathsf{top}(t, D_{i,j})$ is obtained by taking a prefix of size t when sorting the tuples of $D_{i,j}$ from the heaviest to the lightest. Then $C[i, j, k]$ is computed as follows.

$$C[i, j, k] = \begin{cases} 0 & j = 0 \text{ and } k = 0; \\ \infty & j = 0 \text{ and } k > 0; \\ \min\limits_t \Big(C[i, j-1, k-t] + t(k-t)w_\varphi + \sum\limits_{\substack{f \in D_{i,j} \setminus \\ \mathsf{top}(t, D_{i,j})}} w_f \Big) & \text{otherwise.} \end{cases}$$

The correctness of the above computation is due to the definition of the cost in Equation (1). In particular, in the third case, we go over all options for the number t of facts taken from the subblock $D_{i,j}$ and choose an option with the minimum cost. This cost consists of the following components:

- $C[i, j-1, k-t]$ is the cost of the best choice of $k-t$ facts from the remaining $j-1$ subblocks.
- $t(k-t)w_\varphi$ is the cost of the violations in which the jth subblock participates: any combination of a fact from $D_{i,j}$ and a fact from the other subblocks is a violation of φ.
- $\sum_{f \in D_{i,j} \setminus \mathsf{top}(t, D_{i,j})} w_f$ is the cost of removing every fact that is not in $\mathsf{top}(t, D_{i,j})$ from the jth subblock.

This completes the description of the algorithm. From this description, the correctness should be a straightforward conclusion.

4.1 Generalization

We now show how the idea from the previous section can be generalized to some FD sets beyond singletons. An attribute A is an *lhs attribute* of an FD $X \to Y$ if $A \in X$, and it is a *consensus attribute* of $X \to Y$ if $X = \emptyset$ and $A \in Y$ (hence, $X \to Y$ states that all tuples should have the same A value). The simplification step of Algorithm 2 removes an attribute A if for every FD in Δ, it is either an lhs or a consensus attribute. We prove the following.

▶ **Theorem 7.** *Let Δ be a set of FDs. If Δ can be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps, then soft repairing for Δ is solvable in polynomial time.*

Algorithm 2 L/C-Simplify(Δ).

```
1: remove trivial FDs from Δ
2: if Δ is not empty then
3:     find A such that in each FD, A is either an lhs or a consensus attribute
4:     Δ := Δ − A
5: return Δ
```

Note that whenever Δ can be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps, it can also be emptied via $\mathsf{Simplify}(\Delta)$ steps. Indeed, if $\mathsf{L/C-Simplify}(\Delta)$ eliminates the attribute A, then we can take: *(a)* $X = \{A\}$ and $Y = \emptyset$ in Algorithm 1 if A is a consensus attribute of some FD, *or (b)* $X = Y = \{A\}$ if A is an lhs attribute of every FD. This is expected due to Theorems 3 and 7, and the observation of Section 3 that soft-repairing is hard whenever computing a cardinality repair is hard.

▶ **Example 8.** Consider the database and the FD set of our running example (Example 2). This FD set, which we denote here by Δ_1, can be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps, by selecting attributes in the following order:

$$\begin{aligned}
&\{\mathsf{Flight} \to \mathsf{Airline}\,,\ \mathsf{Flight\ Airline\ Date} \to \mathsf{Destination}\}\\
\mathsf{Flight}:&\{\emptyset \to \mathsf{Airline}\,,\ \mathsf{Airline\ Date} \to \mathsf{Destination}\}\\
\mathsf{Airline}:&\{\mathsf{Date} \to \mathsf{Destination}\}\\
\mathsf{Date}:&\{\emptyset \to \mathsf{Destination}\}\\
\mathsf{Destination}:&\{\}
\end{aligned}$$

Hence, Theorem 7 implies that soft repairing can be solved in polynomial time for Δ_1.

Next, consider the FD set Δ_2 consisting of the following FDs: $\mathsf{Flight} \to \mathsf{Airline}$ and $\mathsf{Flight\ Date} \to \mathsf{Destination}$. This FD set is logically equivalent to Δ_1; hence, they both entail the exact same cardinality repairs. However, these sets are no longer equivalent when considering soft repairing. In particular, two facts that agree on the values of the Flight and Date attributes, but disagree on the values of the Airline and Destination attributes, violate only one FD in Δ_1 but two FDs in Δ_2, which affects the cost of keeping these two tuples in the database. In fact, the FD set Δ_2 cannot be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps, as after removing the Flight attribute, no other attribute is either an lhs or a consensus attribute of the remaining FDs. The complexity of soft repairing for Δ_2 remains an open problem. ⌟

Next, we prove Theorem 7 by presenting a polynomial-time algorithm for soft repairing in the case where Δ can be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps. Our algorithm generalizes the idea of the algorithm for a single FD, and we again use dynamic programming.

The main observation is as follows. Let A be an attribute chosen by $\mathsf{L/C-Simplify}(\Delta)$, and let $D_1, \ldots, D_m$ be the maximal subsets of D that agree on the value of A, which we refer to as blocks (w.r.t. A). Two facts from different blocks violate *all* of the FDs wherein A is a consensus attribute and *none* of the FDs wherein A is an lhs attribute. Therefore, to compute the cost of a soft repair, each pair of facts from different blocks is charged with the violation of all FDs wherein A is a consensus attribute. Then, we can remove A from all FDs and continue the computation separately for each block.

Now, let Δ be an FD set that can be emptied via $\mathsf{L/C-Simplify}(\Delta)$ steps, and let $A_1, \ldots, A_n$ be the attributes in the order of such an elimination process. For each $\ell \in \{1, \ldots, n+1\}$, we denote by Δ_ℓ the FD set in line 2 of the ℓth iteration of this execution

(after removing the trivial FDs). Thus, Δ_1 contains every non-trivial FD of Δ, and Δ_{n+1} is empty. We also denote by w_ℓ the total weight of the FDs in Δ_ℓ of which A_ℓ is a consensus attribute (if there are no such FDs, then $w_\ell = 0$).

In the algorithm for a single FD, the recursion steps were with respect to the block D_i (which determines the value of X), and so the value of i was a parameter. Here, we need to maintain the assignment τ to all previously handled attributes, and we use τ and ℓ as parameters. Given $1 \leq \ell \leq n+1$, if τ is an assignment to the attributes $A_1, \ldots, A_{\ell-1}$, then D^τ denotes the database $\sigma_\tau D$ (i.e., the database that contains all the tuples that agree with τ on the values of the attributes $A_1, \ldots, A_{\ell-1}$). We denote by $D^\tau_1, \ldots, D^\tau_{q^\tau_\ell}$ the blocks of D^τ w.r.t. A_ℓ. Moreover, we denote by $\tau \wedge (A_\ell = j)$ the assignment to the attributes $A_1, \ldots, A_\ell$ that agrees with block D^τ_j on the value assigned to A_ℓ and agrees with τ on all other values. We denote by $F[\ell, \tau, j, k]$ an optimal subset of $D^\tau_1 \cup \cdots \cup D^\tau_j$ of size k w.r.t. Δ_ℓ. We also denote by $C[\ell, \tau, j, k]$ the cost of $F[\ell, \tau, j, k]$. According to Equation (1), our goal is to compute $F[1, \emptyset, q^\emptyset_1, k]$ for $k = \operatorname{argmin}_k C[1, \emptyset, q^\emptyset_1, k]$.

We again focus on the computation of $C[\ell, \tau, j, k]$ that can be done as follows.

$$C[\ell, \tau, j, k] = \begin{cases} \sum\limits_{f \in D^\tau \setminus \mathsf{top}(k, D^\tau)} w_f & \ell = n+1, \\ 0 & j = 0, k = 0, \\ \infty & j = 0, k > 0, \\ \min\limits_t \Big(C[\ell, \tau, j-1, k-t] + t(k-t)w_\ell + & \text{otherwise.} \\ \qquad C[\ell+1, \tau \wedge (A_\ell = j), q^{\tau \wedge (A_\ell = j)}_{\ell+1}, t] \Big) & \end{cases}$$

The first line (where $\ell = n+1$) refers to the case where Δ is empty. Since there are no FDs that need to be taken into account, the optimal subset of D^τ of size k consists of the k facts of the highest weight. In the fourth case, we go over all options for the number t of facts taken from the block D^τ_j and choose an option with the minimum cost. This cost consists of the following components:

- $C[\ell, \tau, j-1, k-t]$ is the cost of the best choice of $k-t$ facts from the remaining $j-1$ blocks.
- $t(k-t)w_\ell$ is the cost of the violations in which the jth block participates: any combination of a fact from D^τ_j and a fact from the other blocks $D^\tau_1 \cup \cdots \cup D^\tau_{j-1}$ is a violation of the FDs in which A_ℓ is a consensus attribute.
- $C[\ell+1, \tau \wedge (A_\ell = j), q^{\tau \wedge (A_\ell = j)}_{\ell+1}, t]$ is the cost of the further repairing needed following the elimination of A_ℓ (i.e., repairing with respect to $\Delta_{\ell+1}$) applied to the current block (the t facts from D^τ_j) .

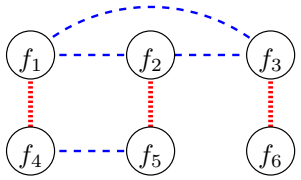
The given recursion can be computed in polynomial time via dynamic programming; thus, this proves Theorem 7.

5 Algorithm for Matching Constraints

In this section, we describe a polynomial-time algorithm for the special case of bipartite matching where the schema is $R(A, B)$ and $\Delta \stackrel{\text{def}}{=} \{A \to B, B \to A\}$. Note that each of the two FDs has a separate weight. In Section 5.1, we extend the algorithm into the more general case of (what we refer to as) a *matching constraint*, where the FD set Δ states two keys that cover all of the attributes. (We give the precise definition in Section 5.1.) We begin by proving the following lemma.

	A	B
f_1	a_1	b_1
f_2	a_1	b_2
f_3	a_1	b_3
f_4	a_2	b_1
f_5	a_2	b_2
f_6	a_3	b_3

(a) Database.



(b) Conflict graph.

Figure 2 A database over $R(A, B)$ and its conflict graph w.r.t. $\{A \to B, B \to A\}$.

▶ **Lemma 9.** *Soft repairing is solvable in polynomial time for* $R(A, B)$ *and* $\Delta = \{A \to B, B \to A\}$.

In the remainder of this section, we assume the input D over $R(A, B)$. We begin with an observation. For $E \subseteq D$ it holds that:

$$\sum_{f \in (D \setminus E)} w_f = \sum_{f \in D} w_f - \sum_{f \in E} w_f$$

Since the value $\sum_{f \in D} w_f$ does not depend on the choice of E, minimizing the value $\left(\sum_{f \in (D \setminus E)} w_f\right) + \left(\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)|\right)$ is the same as minimizing the value $\left(\sum_{f \in E} -w_f\right) + \left(\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)|\right)$. We use the following notation:

$$w_D(E) = \left(\sum_{f \in E} -w_f\right) + \left(\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)|\right)$$

To solve the problem, we construct a reduction to the *Minimum Cost Maximum Flow* (MCMF) problem. The input to MCMF is a flow network $\mathcal{N}$, that is, a directed graph (V, E) with a *source* node s having no incoming edges and a *sink* node t having no outgoing edges. Each edge $e \in E$ is associated with a *capacity* c_e and a *cost* $c(e)$. A flow f of $\mathcal{N}$ is a function $f : E \to \mathbb{R}$ such that $0 \leq f(e) \leq c_e$ for every $e \in E$, and moreover, for every node $v \in V \setminus \{s, t\}$ it holds that $\sum_{e \in I_v} f(e) = \sum_{e \in O_v} f(e)$ where I_v and O_v are the sets of incoming and outgoing edges of v, respectively. A *maximum flow* is a flow f that maximizes the value $\sum_{(s,v) \in E} f(s, v)$, and a *minimum cost maximum flow* is a maximum flow f with a minimal cost, where the cost of a flow is defined by $\sum_{e \in E} f(e) \cdot c(e)$. We say that f is *integral* if all values $f(e)$ are integers. It is known that, whenever the capacities are integral (i.e., natural numbers, as will be in our case), an integral minimum cost maximum flow exists and, moreover, can be found in polynomial time [1, Chapter 9].

From D we construct n instances $\mathcal{N}_1, \ldots, \mathcal{N}_n$ of the MCMF problem, where n is the number of facts in D, in the following way. First, we denote the FD $A \to B$ by φ_1 and the FD $B \to A$ by φ_2. We also denote by $D.A$ the set of values occurring in attribute A in D (that is, $D.A = \{\mathsf{a} \mid \exists f \in D(f[A] = \mathsf{a})\}$). We do the same for attribute B and denote by $D.B$ the set of values that occur in attribute B in D. For each value $\mathsf{a} \in D.A$ we denote by $\#_{D.A}(\mathsf{a})$ the number of appearances of the value a in attribute A (i.e., the number of facts $f \in D$ such that $f[A] = \mathsf{a}$). Similarly, we denote by $\#_{D.B}(\mathsf{b})$ the number of appearances of the value b in attribute B in D. Observe that

$$\mathsf{vio}(D, \varphi_1) = \frac{1}{2} \cdot \sum_{\mathsf{a} \in D.A} [\#_{D.A}(\mathsf{a}) \cdot (\#_{D.A}(\mathsf{a}) - 1)]$$

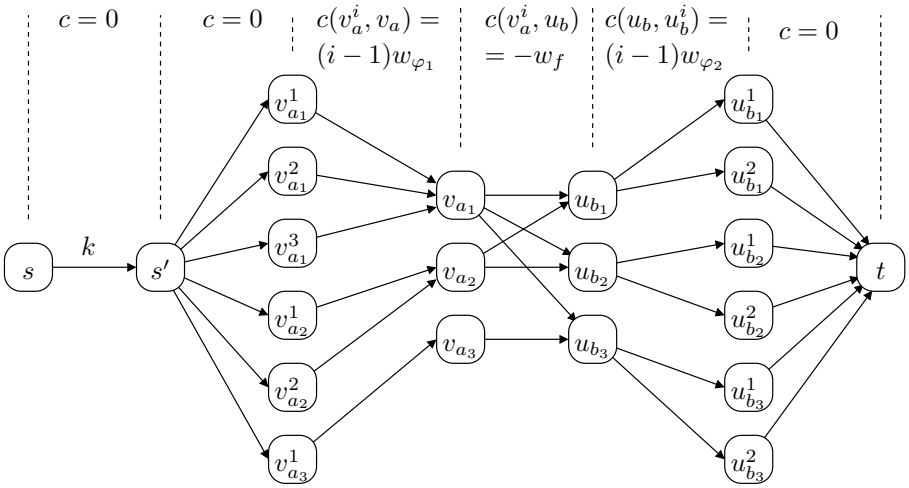


Figure 3 The network $\mathcal{N}_k$ constructed from the database of Figure 2a. The capacity of all edges is 1, except for the edge (s, s') that has capacity k.

since every fact of the form $R(\mathtt{a}, \mathtt{b})$ violates φ_1 with every fact $R(\mathtt{a}, \mathtt{c})$ where $\mathtt{b} \neq \mathtt{c}$. Similarly, it holds that

$$\mathsf{vio}(D, \varphi_2) = \frac{1}{2} \cdot \sum_{\mathtt{b} \in D.B} [\#_{D.B}(\mathtt{b}) \cdot (\#_{D.B}(\mathtt{b}) - 1)]$$

Next, we describe the construction of the network $\mathcal{N}_k$. Our construction for the database of Figure 2a is illustrated in Figure 3. Note that Figure 2b depicts the conflict graph of the database of Figure 2a w.r.t. $\Delta = \{A \to B, B \to A\}$, which contains a vertex for each fact in the database and an edge between two vertices if the corresponding facts jointly violate an FD of Δ. The blue edges in the conflict graph are violations of the FD $A \to B$ and the red edges are violations of the FD $B \to A$.

For each $k \in \{1, \ldots, n\}$ we construct the network $\mathcal{N}_k$ that consists of the set $\{s, s', t\} \cup V \cup A \cup B \cup U$ of nodes where:

- $A = \{v_{\mathtt{a}} \mid \mathtt{a} \in D.A\}$
- $B = \{u_{\mathtt{b}} \mid \mathtt{b} \in D.B\}$
- $V = \{v_{\mathtt{a}}^i \mid \mathtt{a} \in D.A, 1 \leq i \leq \#_{D.A}(\mathtt{a})\}$
- $U = \{u_{\mathtt{b}}^i \mid \mathtt{b} \in D.B, 1 \leq i \leq \#_{D.B}(\mathtt{b})\}$

$\mathcal{N}_k$ contains the following edges:

- (s, s'), with cost $c(s, s') = 0$
- $(s', v_{\mathtt{a}}^i)$ for every $v_{\mathtt{a}}^i \in V$, with cost $c(s', v_{\mathtt{a}}^i) = 0$
- $(v_{\mathtt{a}}^i, v_{\mathtt{a}})$ for every value $\mathtt{a} \in D$, with cost $c(v_{\mathtt{a}}^i, v_{\mathtt{a}}) = (i-1) \cdot w_{\varphi_1}$
- $(v_{\mathtt{a}}, u_{\mathtt{b}})$ for every $\mathtt{a} \in D.A$ and $\mathtt{b} \in D.B$ such that $f = R(\mathtt{a}, \mathtt{b})$ occurs in D, with cost $c(v_{\mathtt{a}}, u_{\mathtt{b}}) = -w_f$
- $(u_{\mathtt{b}}, u_{\mathtt{b}}^i)$ for every value $\mathtt{b} \in D$, with cost $c(u_{\mathtt{b}}, u_{\mathtt{b}}^i) = (i-1) \cdot w_{\varphi_2}$
- $(u_{\mathtt{b}}^i, t)$ for every $u_{\mathtt{b}}^i \in U$, with cost $c(u_{\mathtt{b}}^i, t) = 0$

The capacity of the edge (s, s') is k and the capacity of the other edges is 1. The intuition for the construction is as follows. A network with edges of the form $(v_{\mathtt{a}}, u_{\mathtt{b}})$ that are connected to a source on one side and a target on the other corresponds to a matching, which in turn corresponds to a traditional repair. To allow violations of $A \to B$, we add the vertices $v_{\mathtt{a}}^i$.

The cost of a violation of this FD is defined by the cost of the edges $(v^i_{\mathtt{a}}, v_{\mathtt{a}})$. In particular, if we keep k facts of the form $R(\mathtt{a}, \cdot)$ for some $\mathtt{a} \in D.A$ we pay $\sum_{i=1}^{k}(i-1)w_{\varphi_1}$ for violations of φ_1. We include the vertices $u^i_{\mathtt{b}}$ to similarly allow violations of $B \to A$. The discarding of facts is discouraged by offering gain for the edges $(v_{\mathtt{a}}, u_{\mathtt{b}})$. Finally, to prevent the case where the flow always fills the entire network (which corresponds to taking all facts and paying for all violations), we introduce the edge (s, s') which limits the capacity of the network, and enables us to find the minimum cost flow of a given size k. We will show that for every k, the cost of the solution to the MCMF problem on $\mathcal{N}_k$ will be the cost of the "cheapest" subinstance of D of size k. Hence, the solution to our problem is the cost of the minimal solution among all the instances $\mathcal{N}_1, \ldots, \mathcal{N}_n$.

Given an integral flow f in $\mathcal{N}_k$, the repair $D[f]$ induced by f, is the set of facts $R(\mathtt{a}, \mathtt{b})$ corresponding to edges of the form $(v_{\mathtt{a}}, u_{\mathtt{b}})$ such that $f(v_{\mathtt{a}}, u_{\mathtt{b}}) = 1$. Moreover, given a subinstance E of D of size k, we denote by f_E the integral flow in $\mathcal{N}_k$ defined as follows.

- $f_E(s, s') = k$
- $f_E(s', v^i_{\mathtt{a}}) = 1$ for $1 \leq i \leq \#E.A(\mathtt{a})$ and $f_E(s', v^i_{\mathtt{a}}) = 0$ for $i > \#E.A(\mathtt{a})$ for every $\mathtt{a} \in E.A$
- $f_E(v^i_{\mathtt{a}}, v_{\mathtt{a}}) = 1$ for $1 \leq i \leq \#E.A(\mathtt{a})$ and $f_E(v^i_{\mathtt{a}}, v_{\mathtt{a}}) = 0$ for $i > \#E.A(\mathtt{a})$ for every $\mathtt{a} \in E.A$
- $f_E(v_{\mathtt{a}}, u_{\mathtt{b}}) = 1$ if $R(\mathtt{a}, \mathtt{b}) \in E$ and $f_E(v_{\mathtt{a}}, u_{\mathtt{b}}) = 0$ otherwise
- $f_E(u_{\mathtt{b}}, u^i_{\mathtt{b}}) = 1$ for $1 \leq i \leq \#E.B(\mathtt{b})$ and $f_E(u_{\mathtt{b}}, u^i_{\mathtt{b}}) = 0$ for $i > \#E.B(\mathtt{b})$ for every $\mathtt{b} \in E.B$
- $f_E(u^i_{\mathtt{b}}, t) = 1$ for $1 \leq i \leq \#E.B(\mathtt{b})$ and $f_E(u^i_{\mathtt{b}}, t) = 0$ for $i > \#E.B(\mathtt{b})$ for every $\mathtt{b} \in E.B$

The reader can easily verify that f_E is indeed an integral flow in $\mathcal{N}_k$. Clearly, the value of the flow is k.

We have the following lemma.

▶ **Lemma 10.** *Every integral solution f to MCMF on $\mathcal{N}_k$ satisfies* $\mathsf{cost}(f) = w_D(f[D])$.

Proof. First, note that it cannot be the case that $f(s', v^j_{\mathtt{a}}) = 0$ while $f(s', v^i_{\mathtt{a}}) = 1$ for some $j < i$ and $i \in \{1, \ldots, \#_{D.A}(\mathtt{a})\}$. Otherwise, we can construct a different integral flow f' with $f'(s', v^j_{\mathtt{a}}) = f'(v^j_{\mathtt{a}}, v_{\mathtt{a}}) = 1$, $f'(s', v^i_{\mathtt{a}}) = f'(v^i_{\mathtt{a}}, v_{\mathtt{a}}) = 0$, and $f'(e) = f(e)$ for every other edge e. It holds that $\mathsf{cost}(f') = \mathsf{cost}(f) - c(v^i_{\mathtt{a}}, v_{\mathtt{a}}) + c(v^j_{\mathtt{a}}, v_{\mathtt{a}})$, and since $c(v^i_{\mathtt{a}}, v_{\mathtt{a}}) > c(v^j_{\mathtt{a}}, v_{\mathtt{a}})$ we will have that $\mathsf{cost}(f') < \mathsf{cost}(f)$ in contradiction to the fact that f is a solution to MCMF on $\mathcal{N}_k$. Therefore, for every $\mathtt{a} \in D.A$, if the flow entering the node $v_{\mathtt{a}}$ is ℓ, then $f(s', v^i_{\mathtt{a}}) = f(v^i_{\mathtt{a}}, v_{\mathtt{a}}) = 1$ if $i \leq \ell$ and $f(s', v^i_{\mathtt{a}}) = f(v^i_{\mathtt{a}}, v_{\mathtt{a}}) = 0$ otherwise. Thus, the total cost of the edges of the form $(v^i_{\mathtt{a}}, v_{\mathtt{a}})$ is $\sum_{i=1}^{\ell}[(i-1)w_{\varphi_1}] = \frac{1}{2}\ell(\ell-1)w_{\varphi_1}$. By the definition of $f[D]$, there are $\#_{f[D].A}(\mathtt{a})$ edges of the form $(v_{\mathtt{a}}, u_{\mathtt{b}})$ for which $f(v_{\mathtt{a}}, u_{\mathtt{b}}) = 1$. By the definition of a flow, this is also the flow entering the node $v_{\mathtt{a}}$, and we have that $\ell = \#_{f[D].A}(\mathtt{a})$. We conclude that the total cost of the flow on edges of the form $(v^i_{\mathtt{a}}, v_{\mathtt{a}})$ is

$$\sum_{\mathtt{a} \in f[D].A} \left[\frac{1}{2} \cdot \#_{f[D].A}(\mathtt{a}) \cdot (\#_{f[D].A}(\mathtt{a}) - 1) \cdot w_{\varphi_1}\right] = \mathsf{vio}(f[D], \varphi_1) \cdot w_{\varphi_1}\,.$$

The same argument shows that the total cost of the flow on edges of the form $(u_{\mathtt{b}}, u^i_{\mathtt{b}})$ is $\mathsf{vio}(f[D], \varphi_2) \cdot w_{\varphi_2}$.

Finally, the total cost of the edges of the form $(v_{\mathtt{a}}, u_{\mathtt{b}})$ is $\sum_{g \in f[D]}(-w_g)$ by the definition of $f[D]$ and the construction of the network. We conclude that:

$$\mathsf{cost}(f) = \left(\sum_{g \in f[D]} (-w_g)\right) + \mathsf{vio}(f[D], \varphi_1) \cdot w_{\varphi_1} + \mathsf{vio}(f[D], \varphi_2) \cdot w_{\varphi_2}$$

and $\mathsf{cost}(f) = w_D(f[D])$ by definition. ◀

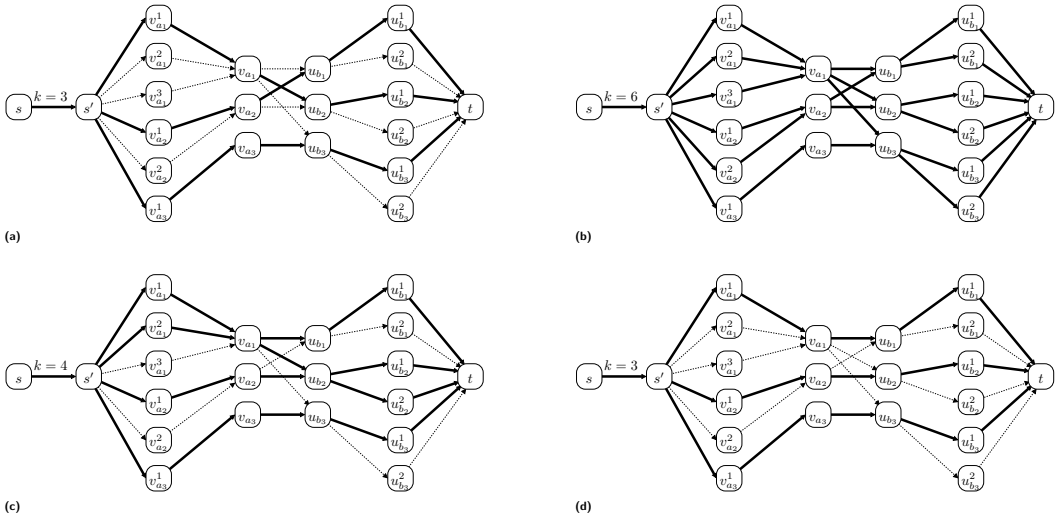


Figure 4 The flow in the network $\mathcal{N}_k$ corresponding to an optimal subset of the database of Figure 2a for different weights.

The next lemma follows straightforwardly from the construction of $\mathcal{N}_k$ and the definition of f_E.

▶ **Lemma 11.** *Every subinstance E of D satisfies* $\mathsf{cost}(f_E) = w_D(E)$.

Now, let E be an optimal subset of D w.r.t. Δ and assume that $|E| = k$. Let f^* be a solution with the minimum cost among all the solutions to MCMF on $\mathcal{N}_1 \ldots, \mathcal{N}_n$. Lemma 11 implies that there is an integral flow f_E in $\mathcal{N}_k$ such that $\mathsf{cost}(f_E) = w_D(E)$. Hence, we have that $\mathsf{cost}(f^*) \leq w_D(E)$. By applying Lemma 10 on f^*, there is another subinstance E' of D such that $w_D(E') = \mathsf{cost}(f^*)$. Since E is an optimal subset, we have that $w_D(E) \leq w_D(E')$. Overall, we have that $\mathsf{cost}(f^*) \leq w_D(E) \leq w_D(E') = \mathsf{cost}(f^*)$, and we conclude that $\mathsf{cost}(f^*) = w_D(E)$. Therefore, by taking the solution with the lowest cost among all solutions to MCMF on $\mathcal{N}_1, \ldots, \mathcal{N}_n$, we indeed find a solution to our problem, and that concludes our proof of Lemma 9.

▶ **Example 12.** Consider again the database of Figure 2a. Assume that:

$$w_{\varphi_1} = w_{\varphi_2} = 2 \qquad w_{f_1} = w_{f_2} = w_{f_3} = w_{f_4} = w_{f_5} = w_{f_6} = 1$$

Since the cost of a violation is "too high" in this case (i.e., it is always cheaper to delete a fact involved in a violation than to keep the violation), an optimal subset in this case is, in fact, an optimal repair in the traditional sense (that is, when the constraints are assumed to be hard constraints). One possible optimal repair in this case is $\{f_2, f_4, f_6\}$. The flow corresponding to this repair in the network $\mathcal{N}_3$ is illustrated in Figure 4a.

Now, assume that:

$$w_{\varphi_1} = w_{\varphi_2} = 1 \qquad w_{f_1} = w_{f_2} = w_{f_3} = w_{f_4} = w_{f_5} = w_{f_6} = 3$$

In this case, the cost of deleting a fact is "too high", since each fact is involved in at most two violations, and the cost of keeping the violation is lower than the cost of removing facts involved in the violation. Therefore, the database itself is an optimal subset, and the corresponding flow in the network $\mathcal{N}_6$ is illustrated in Figure 4b.

As another example, assume that:

$$w_{\varphi_1} = w_{\varphi_2} = 1 \qquad w_{f_1} = w_{f_2} = w_{f_5} = 2, w_{f_3} = w_{f_4} = 1, w_{f_6} = 3$$

Here an optimal subset consists of the facts in $\{f_1, f_2, f_5, f_6\}$, and the corresponding flow in the network $\mathcal{N}_4$ is illustrated in Figure 4c. If we modify the weight of φ_2 and define $w_{\varphi_2} = 4$, while keeping the rest of the weight intact, it is now cheaper to delete the fact f_2 rather than keep the violations it is involved in with f_1 and f_5; hence, an optimal subset in this case is $\{f_1, f_5, f_6\}$, and the corresponding flow in the network $\mathcal{N}_3$ is illustrated in Figure 4d. ⌟

Note that the FD set $\{A \to B\}$ over $R(A, B)$ is in fact a special case of the result of Theorem 14, as we can compute an optimal subset for this FD set using the algorithm described above by defining $w_{B \to A} = 0$. However, this algorithm works only for the case where the single FD is a key and fails to compute the correct solution when the schema contains attributes that do not appear in the FD. The algorithm described in the proof of Theorem 6, on the other hand, can handle this case and does not assume anything about the underlying schema.

5.1 Generalization

We now extend our algorithm beyond bipartite matching to the more general case of a matching constraint. By a "matching constraint" we refer to the case of $\hat{\Delta} = \{X \to Y, X' \to Y'\}$ over a schema $\hat{R}(A_1, \ldots, A_k)$ where $X \cup Y = X' \cup Y' = X \cup X' = \{A_1, \ldots, A_k\}$. An example follows.

▶ **Example 13.** Consider the database of our running example (Figure 1), and the following FDs:

- Flight Airline Date → Origin Destination Airplane,
- Origin Destination Airplane Date → Flight Airline.

The reader can easily verify that these two FDs form a matching constraint. On the other hand, consider the set consisting of the following two FDs:

- Flight Date → Airline Origin Destination Airplane,
- Origin Destination Airplane Date → Flight Airline.

Here, we do not have a matching constraint since while it holds that $X \cup Y = X' \cup Y' = \{\mathsf{Flight}, \mathsf{Airline}, \mathsf{Date}, \mathsf{Origin}, \mathsf{Destination}, \mathsf{Airplane}\}$, the set $X \cup X'$ misses the Airline attribute. ⌟

The generalization of Lemma 9 from $\Delta = \{A \to B, B \to A\}$ over $R(A, B)$ to the general case of a matching constraint is fairly straightforward. Given an input $\hat{D}$ for soft repairing over $\hat{R}$ and $\hat{\Delta}$, we construct an input D over R and Δ by defining unique values $a(\pi_X(\hat{f}))$ and $b(\pi_{X'}(\hat{f}))$ for the projections $\pi_X(\hat{f})$ and $\pi_{X'}(\hat{f})$ over X and X', respectively, of every fact $\hat{f}$ of $\hat{D}$. Then, the database D is simply the set of all the pairs $a(\pi_X \hat{f})$ and $b(\pi_{X'} \hat{f})$ for all facts $\hat{f}$ of D:

$$D \stackrel{\text{def}}{=} \{(a(\pi_X \hat{f}), b(\pi_{X'} \hat{f})) \mid \hat{f} \in \hat{D}\}$$

In addition, we define $w_f \stackrel{\text{def}}{=} w_{\hat{f}}$ whenever $f = (a(\pi_X \hat{f}), b(\pi_{X'} \hat{f}))$ and $w_{A \to B} \stackrel{\text{def}}{=} w_{X \to Y}$ and $w_{B \to A} \stackrel{\text{def}}{=} w_{X' \to Y'}$. Note that the mapping $f \to \hat{f}$ is reversible since $X \cup X' = \{A_1, \ldots, A_k\}$. So, in order to solve soft repairing for $\hat{D}$, we solve it for D and transform every fact f of D into the corresponding fact $\hat{f}$ of $\hat{D}$. We get the following result. The proof is by showing the correctness of the reduction.

▶ **Theorem 14.** *Soft repairing is solvable in polynomial time whenever Δ is a pair of FDs that constitutes a matching constraint.*

Proof. We prove that D has a subset E with $\mathsf{cost}(E \mid D) = k$ if and only if $\hat{D}$ has a subset $\hat{E}$ with $\mathsf{cost}(\hat{E} \mid \hat{D}) = k$. Let E be a subset of D with cost k. Let $\hat{E}$ be a subset of $\hat{D}$ that includes the fact $\hat{f}$ for every $f \in E$. By definition, we have that $\sum_{f \in (D \setminus E)} w_f = \sum_{f \in (\hat{D} \setminus \hat{E})} w_{\hat{f}}$; hence, it is left to show that $\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)| = \sum_{\hat{\varphi} \in \hat{\Delta}} w_{\hat{\varphi}} |\mathsf{vio}(\hat{E}, \hat{\varphi})|$. Let $f, g \in E$ such that $\{f, g\} \not\models (A \to B)$. Hence, it holds that $f[A] = g[A]$ while $f[B] \neq g[B]$. From the construction of D, we have that $\pi_X \hat{f} = \pi_X \hat{g}$, while $\pi_{X'} \hat{f} \neq \pi_{X'} \hat{g}$. Thus, there is an attribute $A_i \in X'$ such that $\hat{f}[A_i] \neq \hat{g}[A_i]$ and since $A_i \notin X$ and $X \cup Y = \{A_1, \ldots, A_k\}$, it holds that $A_i \in Y$. We conclude that $\{\hat{f}, \hat{g}\} \not\models (X \to Y)$. We can similarly prove that if $\{f, g\} \not\models (B \to A)$, then $\{\hat{f}, \hat{g}\} \not\models (X' \to Y')$. Finally, because $w_{A \to B} = w_{X \to Y}$ and $w_{B \to A} = w_{X' \to Y'}$ it holds that $\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)| = \sum_{\hat{\varphi} \in \hat{\Delta}} w_{\hat{\varphi}} |\mathsf{vio}(\hat{E}, \hat{\varphi})|$.

For the other direction, let $\hat{E}$ be a subset of $\hat{D}$, and let E be the subset of D that includes the fact f for every $\hat{f} \in \hat{E}$. It is again straightforward that $\sum_{f \in (D \setminus E)} w_f = \sum_{f \in (\hat{D} \setminus \hat{E})} w_{\hat{f}}$. Now, let $\hat{f}, \hat{g} \in \hat{E}$ such that $\{\hat{f}, \hat{g}\} \not\models (X \to Y)$. We have that $\hat{f}[A_i] = \hat{g}[A_i]$ for every $A_i \in X$; thus, $\pi_X \hat{f} = \pi_X \hat{g}$ and from the construction of D, it holds that $f[A] = g[A]$. On the other hand, the fact that $\hat{f}[A_i] \neq \hat{g}[A_i]$ for some $A_i \in Y$ together with the fact that $X \cup Y = X \cup X' = \{A_1, \ldots, A_k\}$ imply that $\pi_{X'} \hat{f} \neq \pi_{X'} \hat{g}$ and $f[B] \neq g[B]$. Hence, $\{f, g\} \not\models (A \to B)$. We can similarly prove that if $\{\hat{f}, \hat{g}\} \not\models (X' \to Y')$, then $\{f, g\} \not\models (B \to A)$, which again implies that $\sum_{\varphi \in \Delta} w_\varphi |\mathsf{vio}(E, \varphi)| = \sum_{\hat{\varphi} \in \hat{\Delta}} w_{\hat{\varphi}} |\mathsf{vio}(\hat{E}, \hat{\varphi})|$, and the concludes our proof. ◀

6 Conclusions and Open Problems

We studied the complexity of soft repairing for functional dependencies, where the goal is to find an optimal subset under penalties of deletion and constraint violation. The problem is harder than that of computing a cardinality repair, and we have developed two new, nontrivial algorithms solving natural special cases. A full classification of the FD sets remains an open challenge for future research; specifically, the question is what fragment of the positive side of the dichotomy of Livshits et al. [14] remains positive when softness is allowed. We have also shown that the problem becomes tractable if we settle for a 3-approximation.

Open Problems

Several directions are left open for future work. A direct open problem is to characterize the class of tractable FDs via a full dichotomy. The simplest sets of FDs where the complexity of soft repairing is open are the following:

- $\{A \to B, A \to C\}$. Note that this problem is different from $\{A \to BC\}$ that consists of a single FD.
- $\{A \to B, B \to A\}$ in the case where the schema has attributes different from A and B, starting with $R(A, B, C)$.
- $\{\emptyset \to A, B \to C\}$.

The problem is also open for classes of constraints that are more general than FDs, including equality-generating dependencies (EGDs), denial constraints, and inclusion dependencies. Yet, the problem for these types of dependencies is open already in the case of cardinality repairs, with the exception of some cases of EGDs [13]. Another clear direction is that of *update repairs* where we are allowed to change cell values instead of (or in addition to) deleting tuples and where complexity results are known for hard constraints [10, 14].

References

1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

2 Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Discovering denial constraints. *PVLDB*, 6(13):1498–1509, 2013. URL: `http://www.vldb.org/pvldb/vol6/p1498-papotti.pdf`, `doi:10.14778/2536258.2536262`.

3 Carlo Combi, Matteo Mantovani, Alberto Sabaini, Pietro Sala, Francesco Amaddeo, Ugo Moretti, and Giuseppe Pozzi. Mining approximate temporal functional dependencies with pure temporal grouping in clinical databases. *Comp. in Bio. and Med.*, 62:306–324, 2015. `doi:10.1016/j.compbiomed.2014.08.004`.

4 Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by successive approximation. *Math. Oper. Res.*, 15(3):430–466, August 1990. `doi:10.1287/moor.15.3.430`.

5 Teofilo F. Gonzalez, editor. *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007. `doi:10.1201/9781420010749`.

6 Eric Gribkoff, Guy Van den Broeck, and Dan Suciu. The most probable database problem. In *BUDA*, 2014. URL: `http://www.sigmod2014.org/buda`.

7 Dorit S Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on computing*, 11(3):555–556, 1982.

8 Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. TANE: an efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999. `doi:10.1093/comjnl/42.2.100`.

9 Abhay Kumar Jha, Vibhor Rastogi, and Dan Suciu. Query evaluation with soft-key constraints. In *PODS*, pages 119–128, 2008.

10 Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 53–62. ACM, 2009.

11 Weibang Li, Zhanhuai Li, Qun Chen, Tao Jiang, and Zhilei Yin. Discovering approximate functional dependencies from distributed big data. In *APWeb*, pages 289–301, 2016. `doi:10.1007/978-3-319-45817-5_23`.

12 Ester Livshits, Alireza Heidari, Ihab F. Ilyas, and Benny Kimelfeld. Approximate denial constraints. *Proc. VLDB Endow.*, 13(10):1682–1695, 2020. URL: `http://www.vldb.org/pvldb/vol13/p1682-livshits.pdf`.

13 Ester Livshits, Ihab F. Ilyas, Benny Kimelfeld, and Sudeepa Roy. Principles of progress indicators for database repairing. *CoRR*, abs/1904.06492, 2019.

14 Ester Livshits, Benny Kimelfeld, and Sudeepa Roy. Computing optimal repairs for functional dependencies. *ACM Trans. Database Syst.*, 45(1):4:1–4:46, 2020. `doi:10.1145/3360904`.

15 Andrei Lopatenko and Leopoldo E. Bertossi. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 179–193. Springer, 2007.

16 Eduardo H. M. Pena, Eduardo Cunha de Almeida, and Felix Naumann. Discovery of approximate (and exact) denial constraints. *Proc. VLDB Endow.*, 13(3):266–278, 2019. `doi:10.14778/3368289.3368293`.

17 Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006.

18 Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. A formal framework for probabilistic unclean databases. In *ICDT*, volume 127 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

19 Prithviraj Sen, Amol Deshpande, and Lise Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5):1065–1090, 2009.