Combining Quantitative and Logical Data Cleaning

Nataliya Prokoshyna*
University of Toronto
nataliya@cs.toronto.edu

Jaroslaw Szlichta
U. of Ontario Inst. of Tech.
jaroslaw.szlichta@uoit.ca

Fei Chiang
McMaster University
fchiang@mcmaster.ca

Renée J. Miller*
University of Toronto
miller@cs.toronto.edu

Divesh Srivastava
AT&T Labs Research
divesh@research.att.com

ABSTRACT

Quantitative data cleaning relies on the use of statistical methods to identify and repair data quality problems while logical data cleaning tackles the same problems using various forms of logical reasoning over declarative dependencies. Each of these approaches has its strengths: the logical approach is able to capture subtle data quality problems using sophisticated dependencies, while the quantitative approach excels at ensuring that the repaired data has desired statistical properties. We propose a novel framework within which these two approaches can be used synergistically to combine their respective strengths.

We instantiate our framework using (i) metric functional dependencies, a type of dependency that generalizes functional dependencies (FDs) to identify inconsistencies in domains where only large differences in metric data are considered to be a data quality problem, and (ii) repairs that modify the inconsistent data so as to minimize statistical distortion, measured using the Earth Mover's Distance. We show that the problem of computing a statistical distortion minimal repair is NP-hard. Given this complexity, we present an efficient algorithm for finding a minimal repair that has a small statistical distortion using EMD computation over semantically related attributes. To identify semantically related attributes, we present a sound and complete axiomatization and an efficient algorithm for testing implication of metric FDs. While the complexity of inference for some other FD extensions is co-NP complete, we show that the inference problem for metric FDs remains linear, as in traditional FDs. We prove that every instance that can be generated by our repair algorithm is set-minimal (with no unnecessary changes). Our experimental evaluation demonstrates that our techniques obtain a considerably lower statistical distortion than existing repair techniques, while achieving similar levels of efficiency.

*Supported in part by NSERC BIN

Proceedings of the VLDB Endowment, Vol. 9, No. 4

1. INTRODUCTION

Two major trends in data cleaning have emerged. The first is a logical approach to cleaning; the second a quantitative or statistical one. In logical data cleaning, error detection and repair is typically performed using declarative cleaning programs [12]. For example, if a cleaning program asserts that a sensor should have a single reading at any time point, it may also contain a resolution strategy that replaces multiple readings with their average. In constraint-based cleaning, data dependencies are used to detect data quality problems. Data that is inconsistent with respect to the constraints can be repaired by finding a minimal set of changes that fix the errors [7, 15]. An important advantage of such approaches is that they are able to find subtle data quality problems using sophisticated dependencies. Repair is typically done based on minimizing the number or cost of changes needed to create a consistent database.

In quantitative data cleaning, there has been considerable work on outlier detection for quantitative data (both univariate and multivariate or relational data) [14]. This has been complemented with work on pairing repair strategies with detection which considers multiple types of errors (for different types of data) [3]. An important insight in this latter work is that the cleaned data should be of higher data quality than the original uncleaned data. To quantify this, Berti-Equille et al. [3] propose cleaning techniques that ensure the cleaned data is statistically close to an *ideal* dataset. Dasu and Loh [9] coined the term *statistical distortion* for the distance between the distribution of a (cleaned) dataset and a user-defined *ideal* dataset. This work proposes statistical distortion as a way to (*post facto*) evaluate (and compare) different cleaning strategies. However, a quantitative repair is not guaranteed to be logically consistent.

There has, to date, been little work combining the insights of these two separate threads of research. Yakout et al. [25] propose a cleaning technique that separates detection from repair. Detection may be done using clean master data, using constraints or using quantitative outlier detection. Repair on the detected (potential) errors is done using quantitative cleaning. However, unlike our approach they do not make use of integrity constraints to obtain a finer-grained view of what data needs to be repaired, ensure the repair is a consistent database, or guarantee that only minimal (necessary) changes are done.

In contrast, our technique combines quantitative and logical approaches. We propose a new constraint-based cleaning strategy in which we use statistical distortion during cleaning to ensure the chosen (minimal) repair is of high quality.

TID	Source	Person	Position	Organization	Latitude	Longitude
t_1	$S3$	NP	Student	Univ of Toronto	43.662892	-79.395656
t_2	$S3$	JS	Student	Univ of Toronto	**-33.0142425**	**151.5818976**
t_3	$S2$	NP	Student	University of Toronto	43.943445	-78.895452
t_4	$S3$	FC	Student	Univ of Toronto	**40.4587165**	**-80.6088795**
t_5	$S2$	FC	Faculty	McMaster Univ	43.260879	-79.919225
t_6	$S2$	RJM	Faculty	Univ of Toronto	43.662892	-79.395656
t_7	$S1$	RJM	Faculty	Univ of Toronto	43.66	-79.40
t_8	$S2$	DS	Manager	AT&T Labs Research	40.669550	-74.636399
t_9	$S1$	DS	Manager	Shannon Labs	40.67	-74.63
t_{10}	$S3$	DS	Manager	AT&T Labs Research	**30.391161**	**-97.751548**

$M1$: Person, Position $\mapsto$ Organization ($\theta = 6$) $M2$: Organization $\mapsto$ Latitude ($\theta = 0.01$) $M3$: Organization $\mapsto$ Longitude ($\theta = 0.01$)

Table 1.1: A dirty relation & data quality constraints.

TID	Source	Person	Position	Organization	Latitude	Longitude
t'_2	$S3$	JS	Student	Univ of Toronto	43.66	-79.40
t'_4	$S3$	FC	Student	Univ of Toronto	43.662892	-79.395656
t'_{10}	$S3$	DS	Manager	AT&T Labs Research	40.669550	-74.636399

Table 1.2: Repaired tuples.

1.1 Cleaning Example

Consider the relation in Table 1.1. The data is integrated information about employees, their positions in organizations, and the locations (latitude, longitude) of the organizations, from multiple sources. For our data quality constraints, we make use of *metric functional dependencies* (metric FDs) [16]. Metric FDs generalize traditional FDs to permit some variation in the values of attributes that appear in their consequent. They are appropriate in domains where only a large variation in values indicates a real semantic difference. In each of the three dependencies (Table 1.1), the value of θ indicates how much variation is permitted. For values in the `Latitude` and `Longitude` attributes, we use a metric that is the difference in degrees; $\theta = 0.01$ is approximately 1km. For values in the `Organization` attribute, we use a metric that is the edit distance (with $\theta = 6$). For example, the edit distance between `Univ of Toronto` and `University of Toronto` is six, therefore these two values satisfy the threshold. Additionally, we assume a preprocessing step is performed that takes into account organizational synonyms. Since `AT&T Labs Research` and `Shannon Labs` are synonyms we use their canonical name, for instance, `AT&T Labs Research` (Section 2.1).

The metric FD $M1$ indicates that if two tuples share the same `Person` and `Position` values, then the edit distance between their `Organization` values should 6 or less (indicated by the $\theta = 6$ in Table 1.1); this variation may be permitted due to conventions in different sources. Hence, t_1 and t_3 do not violate $M1$. Assuming that `AT&T Labs Research` and `Shannon Labs` are synonyms with a canonical name `AT&T Labs Research`, then normalized tuples t_8 and t_9 (and our entire example relation) satisfy $M1$.

The metric FDs $M2$ and $M3$ indicate that if two tuples share the same `Organization` value (i.e., their edit distance is 0), then their `Latitude` and `Longitude` values, respectively, should differ by no more than 0.01 degree; this variation may be permitted due to differences in GPS measurements or in conventions about the exact location of an organization. Hence, tuples t_6 and t_7 together satisfy these dependencies, as do tuples t_8 and t_9, while t_1 and t_2 do not.

Consider a purely logical approach to cleaning, using a minimal repair. There are different notions of minimality in the literature, but for illustration, consider only modifications of attribute values (i.e., no deletion or insertion of tuples). A repair is minimal if we cannot undo any modification and still have a relation that is consistent; as we explain in Section 2, this is called a *set-minimal* repair [5].

Consider repairing the inconsistencies on metric FDs $M2$ and $M3$ among tuples for the organization `Univ of Toronto` (namely, t_1, t_2, t_4, t_6 and t_7). A set-minimal repair could change the `Latitude` values of t_2 and t_4 to be within 0.01 degrees of both 43.66 and 43.662892, and the `Longitude` values of t_2 and t_4 to be within 0.01 degrees of both -79.395656 and -79.40. But set-minimality does not tell us which values in these ranges to pick. We could use a measure like support [8] to change a value to the most frequent value (in this case, 43.662892 for `Latitude` and -79.395656 for `Longitude`). Or we could use Winsorization [9] to change a value to the closest value that is consistent with the other tuples (in this case, both the `Latitude` values -33.0142425 in t_2 and 40.4587165 in t_4 would change to 43.66). However, such approaches can lead to statistical drift, with frequent values being overly represented in the cleaned relation, or outlier erroneous values (like -33.0142425) pulling the data distribution too far in the direction of the outliers.

Alternatively, we could change the `Organization` of t_2 and t_4 to be *different* from that of t_1, t_6, t_7. This change would be minimal as well, and would involve changing only two attribute values instead of four values as in the case of changing `Latitude` and `Longitude` values of t_2 and t_4. But which repair is better? And what values (in the specified ranges) should we pick for the repairs? Some approaches use measures like support or co-occurrence of values within the dirty relation itself to pick among alternate minimal repairs. We propose instead to pick a repair that minimizes *statistical distortion* with respect to an ideal relation [3, 9, 25].

To illustrate why the strategy to minimize statistical distortion can be desirable, consider $M2$ and $M3$ in our example. Instead of creating a potential statistical bias by repairing the `Latitude` and `Longitude` values of t_2 and t_4 to the most frequent or the closest values, or changing the `Organization` values of t_2 and t_4 to some different values, our approach will use an ideal relation (Section 2) as a guide, and pick repair values so as to mimic the ideal distribution $D_{\mathcal{I}}$ as close as possible.

Most existing repair approaches use a notion of minimality (minimizing the number of changes) [5] or minimal cost repair [4, 7, 15] to decide on values to use in repairing errors.

Their objective function is based on the number or cost of repairs with respect to the original dirty dataset. Like these existing approaches, we could define $D_{\mathcal{I}}$ as the original dirty dataset. The difference would be the objective function – they minimize counts or cost (where cost is defined with respect to a distance function over attribute values), while we minimize statistical distortion. A benefit of our approach however (one that has been observed by others [3, 9, 25]) is that we have freedom in defining $D_{\mathcal{I}}$. It can be the original dirty data or it can be a clean subset of the data. A user can decide if the evidence they want to use in cleaning is best captured by the original data, or only the data without the errors (since these errors may change the distribution). Alternatively, and most realistically, in a continuous data cleaning process [23], a user can use a previously cleaned and verified dataset as $D_{\mathcal{I}}$.

Our approach is suitable for applications where errors are introduced in a relatively independent manner and the amount of error remains small relative to the data size or to a previously cleaned $D_{\mathcal{I}}$. Such datasets are common in practice (e.g., sensor based applications, flight arrival times, GPS transit tracking) where unexpected events (e.g., dropped data or delays) can lead to errors, but the data is expected to conform to a (known) distribution of expected events. In contrast, when errors are introduced systematically, for example by a faulty information-extraction process, then other approaches, like the Data X-Ray [24], may be used.

In our example, if the ideal relation is the user-specified set of tuples $t_1, t_3, t_5, t_6, t_7, t_8$ and t_9 (which in this case is a maximal consistent subset of the dirty data), the distributions of values for `Organization`, `Latitude` and `Longitude` in the ideal relation leads to a repair of t_2, t_4 and t_{10} shown in Table 1.2. Note that distributions of attribute values in the cleaned relation $t_1, t'_2, t_3, t'_4, t_5, t_6, t_7, t_8, t_9, t'_{10}$ is statistically closer or equal to the ideal relation than other modification-only repairs that could be performed.

1.2 Contributions

• **A new framework that combines the benefits of logical and quantitative data cleaning.** A *statistical-distortion minimal repair* is a consistent relation that both differs set-minimally from the original dirty relation and that differs minimally, in terms of statistical distortion from an ideal relation. We have chosen to use metric FDs (a strict superset of FDs) to capture data quality requirements, because of their importance in representing the semantics of integrated data (where small differences in a quantitative attribute may not indicate a semantic difference). We use metric FDs to identify errors in the data, and we propose statistical repairs to resolve these errors.

• **An algorithm that efficiently and effectively computes a set-minimal repair with low statistical distortion.** We show that the problem of computing statistical distortion minimal repairs is NP-hard. Our algorithm uses the Earth Mover's Distance (EMD) to calculate the statistical distortion. Computing the EMD is well-known to be computationally expensive [17, 18]. Hence, we propose several optimizations via sampling that significantly improve the performance of our algorithm without compromising the quality of our repairs. These include a pruning strategy that removes low frequency values (at the tail of the distribution) to reduce the search space of repairs since these values are unlikely to be chosen in a statistical repair.

• **The foundations of how *metric FDs* can be used for data cleaning.** While FDs and extensions including conditional FDs [6] and matching dependencies [10] have been used extensively for data cleaning, ours is the first application of metric FDs to cleaning. To permit their use in cleaning, we must understand the inference problem for metric FDs. To this end, we present a sound and complete axiomatization for metric FDs. In contrast to another FD extension that uses similarity, called differential dependencies [20], for which inference is co-NP-complete, we show that the inference problem for metric FDs remains linear, as in traditional FDs. Our cleaning solution includes an implementation of our inference algorithm to compute the minimal cover of a set of metric FDs, and the closure of a set of attributes under a set of metric FDs. As an optimization, our algorithm computes EMD only over attribute closures rather than over the whole relation.

• **An experimental evaluation showing the performance and effectiveness of our techniques using real and synthetic data.** We experiment with different error injection strategies (resulting in different relational data distributions), and show that our framework achieves small statistical distortion when compared against an ideal data distribution. Finally, we show that our algorithm is able to achieve improved (lower) statistical distortion over a recent logical data repair solution [8].

2. PROBLEM DEFINITION

To combine logical and quantitative data cleaning, we will use a set of constraints M to both identify errors and to define a space of possible corrections (repairs). Following common practice in quantitative data cleaning [9, 3, 25], we make use of an ideal distribution. We use $D_{\mathcal{I}}$ to measure the quality of a repair, that is, how well a repair preserves the desired distribution of the data. Abstractly, we use the following problem definition.

Problem Definition Given a set of constraints M, a dataset D where $D \not\models M$, and a dataset reflecting the desired data distribution $D_{\mathcal{I}}$, find a repair D_r where $D_r \models M$ and the statistical distortion between D_r and $D_{\mathcal{I}}$ is minimized.

We now give the specifics of the constraint language we consider, the class of repairs, and how we will measure distance between two relations.

2.1 Logical Foundations

To begin, our constraint language will include functional dependencies (FDs), the most common constraint used in data cleaning. In addition, we have chosen to consider metric functional dependencies, a strict superset of FDs [16]. While the verification problem (determining if $D \models M$), for metric FDs has been studied, their use in data cleaning to guide data repair has not been studied. Metric FDs are defined over a relation where for each attribute, we have a metric m and threshold parameter $\theta \geq 0$.

Definition 2.1. (metric schema)
A **metric schema** *is a set of attributes* $\boldsymbol{R}$ *where for every attribute* $A \in \boldsymbol{R}$*, we have metric* m_A *and threshold* $\theta_A \geq 0$*. Each metric* m_A *satisfies standard properties: symmetry, triangle inequality and identity of indiscernibles (*m_A*(a, b) = 0 iff a = b). Let* $X \subseteq \boldsymbol{R} = \{A_1, ..., A_n\}$*. For two tuples s, t over* $\boldsymbol{R}$*, we write* $s[X] \approx_{m_X, \Theta_X} t[X]$ *to mean* $s[A_1] \approx_{m_{A1}, \theta_{A1}} t[A_1]$*, ...,* $s[A_A] \approx_{m_{An}, \theta_{An}} t[A_n]$*.*

A metric FD generalizes FDs to permit slight variations in a functionally determined attribute. Note that if $\theta = 0$ then a metric FD is an FD.

DEFINITION 2.2. (metric FD)
Given a relation D over a metric schema $\boldsymbol{R}$, let X and Y be sets of attributes in $\boldsymbol{R}$, $\boldsymbol{m}_Y$ and Θ_Y be the metrics and thresholds for all the attributes in Y. Then, $X \mapsto Y$ denotes a **metric FD**. *A relation D satisfies $X \mapsto Y$ ($D \models X \mapsto Y$), iff for all tuples s, $t \in D$, $s[X] = t[X]$ implies $s[Y] \approx_{\boldsymbol{m}_Y, \Theta_Y} t[Y]$.*

To permit attributes to contain synonyms (like the *Organization* attribute in our running example), we assume that each attribute value has a canonical name (in our example, `AT&T Labs Research` and `Shannon Labs` have a canonical name `AT&T Labs Research`). Therefore, for any relation D, there is a function $normalize(D)$ that replaces attribute values with canonical names. So when we write $D \models M$, we mean $normalize(D) \models M$.

EXAMPLE 2.3. *Let $D = \{t_1, ..., t_{10}\}$ from Table 1.1 and let $M = \{M_1, M_2, M_2\}$. As we argued in Section 1 $D \not\models M$, more specifically once we normalize D, M_1 is satisfied, but M_2 and M_3 are not. However, if we remove t_2, t_4 and t_{10}, then the remaining seven tuples do satisfy all constraints since the difference in* Organization *values for tuples with the same* Person *and* Position *values is at most 6 and the difference in* Latitude *and* Longitude *values for tuples with the same* Organization *is at most 0.01.*

DEFINITION 2.4. (consistent relation)
Given M, a set of metric FDs, a relation D is **consistent** *iff $D \models M$. Otherwise, D is inconsistent (or dirty).*

We define repairs using value modification only, that is, attribute values of a tuple may be changed but tuples may not be inserted or deleted.

DEFINITION 2.5. (repair)
A repair of an inconsistent relation D_d is a consistent relation D_r that can be created from D_d using a set of value modifications $\mathcal{V}$.

Obviously, not all repairs are equally valuable. Changing the value of every attribute to 1 in our running example creates a repair. But such a repair is intuitively less desirable than the repair we depict in Table 1.2. Hence, there are several well-known notions of minimal repairs for FDs that are based on minimizing the number or cost of the changes made to the data. A well-known notion of minimality is *cardinality-minimal repairs* [5, 7, 15]. A repair is cardinality-minimal if the number of changes ($|\mathcal{V}|$) is the smallest possible among all possible repairs. A less restrictive notion is *set-minimal* repairs. A repair D_r created by a set of value modifications is set-minimal if no subset of the changed values in D_r can be reverted to their original values without violating M [1, 5]. Cardinality-minimal repairs are set-minimal, but not vice versa.

DEFINITION 2.6. (set-minimal repair)
A repair D_r created by a set of changes $\mathcal{V}$ of an inconsistent relation D_d is set-minimal iff there is no repair $D_{r'}$ that can be created by a strict subset $\mathcal{V}' \subset \mathcal{V}$.

In set-minimal repairs, every change is necessary. Hence, the decision of whether to change a value is driven exclusively by the constraints. No value is changed unless it is necessary to restore consistency. However, set-minimality leaves some freedom (more freedom than cardinality-minimality) in selecting what values to use for modification, i.e., how to repair. We will use statistical distortion to guide this choice and decide which set-minimal repair to choose.

2.2 Statistical Foundations

We assume a relation $D_{\mathcal{I}}$ that follows the desired data distribution. Given a set of candidate repairs, we will consider one repair better than another if it has lower statistical distortion from $D_{\mathcal{I}}$. Let $MD(D)$ be the multi-dimensional distribution of a relation D.

DEFINITION 2.7. (statistical distortion)
Given relations D_1 and D_2, the statistical distortion *(SD) is defined as a distance between the distributions of D_1 and D_2, denoted as distance($MD(D_1)$, $MD(D_2)$).*

For simplicity, we write *distance*(D_1, D_2) to mean *distance*($MD(D_1)$, $MD(D_2)$). Following Dasu and Loh [9], to calculate the distance between distributions, we use the Earth Mover's distance (EMD). EMD is a distance function that measures the dissimilarity of two histograms. In our work, this is based on the frequencies of the unique values in the dataset, often computed over a subset Z of the attributes in $\mathbf{R}$. The set of *bins* in the histogram is the set of tuples in $\Pi_Z(D)$. The *weight*, w_{p_i}, of a bin p_i is the relative number of occurrences of this value in D (meaning $|\{t|\Pi_Z(t) = p_i\}|$ divided by $|D|$, where t is a tuple in D).

DEFINITION 2.8. (Earth Mover's Distance (EMD))
Given two relational histograms, let $P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$ and $Q = \{(q_1, w_{q_1}), ..., (q_n, w_{q_n})\}$, each having m and n bins respectively. Define a cost matrix C, where $c_{i,j}$ is a measure of dissimilarity between p_i and q_j, that models the cost of transforming p_i to q_j, and a flow matrix F where $f_{i,j}$ indicates the flow capacity between p_i and q_j. EMD is defined in terms of an optimal flow that minimizes

$$d(P, Q) = \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} * c_{ij} \tag{1}$$

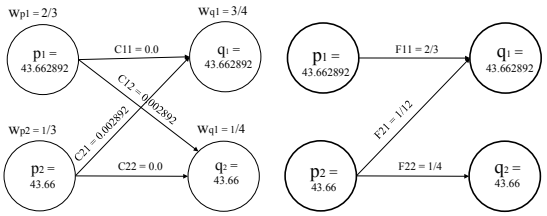
The EMD is defined as follows:

$$EMD(P, Q) = \min_{F} d(P, Q) \tag{2}$$

subject to the following constraints:

$$\begin{aligned} &\forall i \in [1, m], \forall j \in [1, n] : f_{i,j} \geq 0, \\ &\forall i \in [1, m] : \sum_{j=1}^{n} f_{i,j} = w_{p_i}, \\ &\forall j \in [1, n] : \sum_{i=1}^{m} f_{i,j} = w_{q_j}, \end{aligned} \tag{3}$$

Equations 1-3 in Definition 2.8 are defined based on changing the distribution of P to Q (and not vice versa) and limit the amount of change that can be done to the bins in P and Q according to their respective weights. For two distributions P and Q, EMD computes the distance between every i-th bin in P and every j-th bin in Q. This is known as the cross-bin difference, and can be seen in Figure 2.1(a) where an edge exists between every pair of nodes in the graph. Considering cross-bin differences makes EMD less sensitive



(a) Flow graph. (b) Minimum cost flow.

Figure 2.1: The flow network.

to the positioning of bin boundaries (note that a bin need not contain only one value p_i, but may in general contain many consecutive values).

We now define **statistical-distortion minimal repair**.

DEFINITION 2.9. (statistical-distortion minimal repair) *Given an ideal relation $D_\mathcal{I}$ and a relation D_d that is inconsistent with respect to a set of metric FDs M, a* **statistical-distortion (SD) minimal repair** *is a set-minimal repair D_r for which EMD(D_r, $D_\mathcal{I}$) is minimal.*

Despite the fact that we are relaxing minimality to consider set-minimal repairs which are easier to find than cardinality-minimal repairs (the problem of finding a cardinality-minimal repair for FDs is NP-hard [7, 15]), the problem of finding a SD minimal repair remains NP-hard.

THEOREM 2.10. (complexity) *The problem of finding a SD minimal repair is NP-hard.*

3. STATISTICAL REPAIR MODEL

We propose a greedy algorithm that finds a maximal consistent subset of a dirty relation D_d, and searches for a minimal repair of each inconsistent tuple one at a time. We select among alternate tuple repairs by minimizing statistical distortion. We first discuss the main structure of the algorithm, then discuss some additional ways of making the computation more efficient over large relations.

3.1 Algorithm for Generating Repairs

Let D_d be a dirty relation and let $D_c \subset D_d$ be a maximally consistent set of tuples from D_d, meaning there is no tuple $t \in D_d \setminus D_c$ such that $\{t\} \cup D_c$ is consistent. We call $D_u = D_d \setminus D_c$ the unresolved data.

For each tuple $t_d \in D_u$, we compute a set of possible modifications (tuple repairs). Each repair modifies attribute values in t_d to create a candidate repaired tuple t_r where $\{t_r\} \cup D_c$ is consistent. Among the possible tuple repairs for t_d, we select the tuple repair that minimizes statistical distortion from $D_\mathcal{I}$. To ensure we generate a set-minimal repair, we ensure that t_r is minimal, meaning that we cannot revert any value in t_r to the value in t_d and still have a tuple repair. Our algorithm assumes that the set of metric FDs M is a minimal cover meaning, among other things, that each constraint has the form $X \mapsto A$ (θ_A) where A is a single attribute. Algorithms to compute a minimal cover for FDs are well-known. We show in Section 4 how to compute a minimal cover for metric FDs. Note that our algorithm is greedy. We repair tuples one at a time (without backtracking) and minimize the distortion of the current clean subset. Of course, considering tuples in different orders (or using different maximally consistent subsets of tuples to start our algorithm) may lead to different repair choices. We empirically evaluate the influence of these choices on the quality of our solution in Section 5. Our algorithm is guaranteed to create a set-minimal repair and strives to minimize distortion, but is not guaranteed to find a statistical-distortion minimal repair.

Algorithm 1 StatisticalDistRepair

Input: M (a minimal cover of a set of metric FDs), inconsistent relation $D_d \not\models M$, maximal clean subset $D_c \subseteq D_d$, ideal relation $D_\mathcal{I}$
Output: Repaired relation D_r of D_d

1: $D_u = D_d - D_c$ (unresolved set)
2: **while** $D_u \neq \{\}$ **do**
3: select $t_d \in D_u$; $D_u = D_u - \{t_d\}$
4: $M[t_d] = \{m \in M | \{t_d\} \cup D_c \not\models m\}$ (deps violated by t_d)
5: $U = \cup_{X \mapsto A \in M[t_d]} X$ (antecedents in violated deps)
6: $V = \cup_{X \mapsto A \in M[t_d]} A$ (consequents in violated deps)
7: $Z = \cup XA \in M$ (union of all attributes in M)
8: Candidates = $\emptyset$ (find consequent repair candidates)
9: **for all** $v \in \Pi_V(\sigma_{U=t_d[U]}(D_c))$ **do**
10: $t_v = t_d$; $t_v[V] = v$
11: **if** $\{t_v\} \cup D_c \models M$ **then**
12: Candidates = Candidates $\cup \{t_v\}$
13: **end if**
14: **end for**
15: **if** Candidates $\neq \emptyset$ **then**
16: $t_{cons} = argmin_{t \in Candidates} EMD(t \cup D_c, D_\mathcal{I})$
17: **else**
18: $t_{cons} = \emptyset$
19: **end if**
20: Candidates = $\emptyset$ (find antecedent repair candidates)
21: **for all** $u \in \Pi_U(V = \sigma_{t_d[V]}(D_c)))$ **do**
22: $t_u = t_d$; $t_u[U] = u$
23: **if** $\{t_u\} \cup D_c \models M$ **then**
24: Candidates = Candidates $\cup \{t_u\}$
25: **end if**
26: **end for**
27: **if** Candidates $\neq \emptyset$ **then**
28: $t_{ante} = argmin_{t \in Candidates} EMD(t \cup D_c, D_\mathcal{I})$
29: **else**
30: $t_{ante} = \emptyset$
31: **end if**
32: **if** $(t_{cons} = \emptyset) \wedge (t_{ante} = \emptyset)$ **then**
33: Candidates = k tuples in D_c closest to $t_d[Z]$
34: $t_r = argmin_{t \in Candidates} EMD(t \cup D_c, D_\mathcal{I})$
35: **else**
36: **if** $(t_{ante} = \emptyset) \vee (t_{cons} \neq \emptyset \wedge EMD(t_{cons} \cup D_c, D_\mathcal{I}) < EMD(t_{ante} \cup D_c, D_\mathcal{I}))$ **then**
37: $t_r = t_{cons}$
38: **else**
39: $t_r = t_{ante}$
40: **end if**
41: **end if**
42: $D_c = D_c \cup \{$ set-minimal(t_r)$\}$
43: **end while**
44: **return** $D_r = D_c$

3.2 Tuple Repairs

To generate tuple repairs for a tuple t_d, we must consider that t_d may violate many constraints in M. Let $M[t_d] = \{m \in M | \{t_d\} \cup D_c \not\models m\}$ be the set of dependencies that would be violated if t_d were added unmodified to D_c. Let U be the union of antecedents in $M[t_d]$. Let V be the union of the consequents in $M[t_d]$. For each tuple, we first consider modifying attributes in V to create consequent-repairs (Lines 9-14 of Algorithm 1), then we consider modifying

TID	CEO	Organization	Longitude
t_1	Stephenson	AT&T	-74.636399
t_2	Stephenson	AT&T	-74.636399
t_3	Stephenson	IBM	**-97.751548**
t'_r	Stephenson	AT&T	**-97.751548**
t''_r	Stephenson	AT&T	-74.636399

$M1'$: CEO $\mapsto$ Org ($\theta = 0$) $M2'$: Org $\mapsto$ Longitude ($\theta = 1$)

Table 3.1: Repairing both sides to satisfy constraints.

attributes in U to create antecedent-repairs (Lines 21-26). For consequent-repairs, we consider modifying $t_d[V]$ (the tuple projected on the attributes of V) to each value in $\Pi_V(U = \sigma_{t_d[U]}(D_c))$. Intuitively, this means that we repair the tuple to have the same consequent as some other tuple in D_c that shares its antecedent. This defines a set of candidate consequent-repairs. We keep only those repairs t_r such that $\{t_r\} \cup D_c \models M$. We then compute the repair that minimally distorts D_c among all these candidate repairs ($argmin_{t_r} EMD(\{t_r\} \cup D_c, D_{\mathcal{I}})$). (Note in the rest of the paper we omit set notation and write $t_r \cup D_c$ for simplicity.)

We do the same for antecedent-repairs. Specifically, we consider modifying $t_d[U]$ to each value in $\Pi_U(V = \sigma_{t_d[V]}(D_c))$. This defines a set of candidate antecedent-repairs. Again, we keep only those repairs t_r such that $t_r \cup D_c \models M$ and again find one repair among these that minimally distorts D_c from $D_{\mathcal{I}}$. If either a best consequent or a best antecedent repair is found, we select the one with minimum distortion. Throughout, we break ties arbitrarily.

EXAMPLE 3.1. *Continuing our example from Section 1, Table 1.1, we may select the following maximally clean subset: $D_c = \{t_1, t_3, t_5, t_6, t_7, t_8, t_9\}$. Suppose we select t_4 as t_d which violates M_2 and M_3. There are two possible consequent-repair candidates for t_4, for each we would change only the* `Lat` *and* `Long` *values of t_4.*

i) t'_4[`Lat`, `Long`] = [43.662892, −79.395656] *(uses t_1, t_6)*

ii) t''_4[`Lat`, `Long`] = [43.66,−79.40] *(uses t_7)*

We choose the lower distortion repair by computing EMD($t'_4 \cup D_c, D_{\mathcal{I}}$) and EMD($t_4'' \cup D_c, D_{\mathcal{I}}$). We also compute the set of candidate antecedent-repairs. Antecedent-repairs in this example change the `Org` *value of t_4 to be an organization that is located at* `Lat` =40.4587165 *and* `Long` = −80.6088795. *However, there are no such organizations in the clean dataset (no antecedent-repair candidates).*

If we fail to find a valid antecedent or consequent-repair, then we consider modifying values in Z, where Z is all attributes in M (we call this a both-repair, see Lines 32–34 of Algorithm 1). Following the logic we used for other repairs, we could modify $t_d[Z]$ to each value in $\Pi_Z(D_c)$. However, this leads to a large number of candidate tuple repairs and they will all be consistent wrt D_c. We limit this set to k tuples that are the closest to t_d (meaning the distance $m_Z(\Pi_Z(t_d), \Pi_Z(t_r))$ is minimum). Again, we keep the repair that minimally distorts D_c with respect to $D_{\mathcal{I}}$.

EXAMPLE 3.2. *Consider the new example in Table 3.1, which violates dependency $M1'$ and satisfies $M2'$. If we attempt to repair t_3*[`Organization`]=[*AT&T*] *the repair (t'_r) would violate $M2'$. Therefore, there is no suitable antecedent or consequent repair for this tuple, we can only repair both sides. A both-repair is t''_r.*

As a final step, we verify that our recommended repairs are set-minimal (Line 42). For a chosen tuple t_r, we iterate through all combinations of attributes in $W \subseteq Z$, ordered by size from large to small, we revert the current values in $t_r[W]$ back to their original values, to obtain a tuple t'_r. If we find a subset W such that $(D_c \cup t'_r) \models M$, we replace t_r with t'_r. Once this process terminates we add the resulting set-minimal tuple to D_c. The reversion step can be performed efficiently, since it is done at the schema level over the attributes in M, which is smaller than the size of the data. We experimentally verified that the reversion step takes less than 10% of the total running time of our algorithm. Note however, that performing this check for each candidate repair could be expensive, hence we only perform the check once for each dirty tuple t_d. Note that set-minimal(t_r) may add more (or possibly less) distortion than t_r, but we have not found this effect to be significant. The following theorem states that Algorithm 1 produces a set-minimal repair, after iterating through all the tuples t_d.

THEOREM 3.3. (Set-minimal repair algorithm) *Algorithm 1 creates a set-minimal repair of the inconsistent relation D_d over the set of metric FDs M and has worst case complexity $\mathcal{O}(n^3 log^2\ n)$.*

Note that Algorithm 1 does not necessarily create a statistical-distortion minimal repair. Each inconsistent tuple is only considered once (hence we go through the *while* loop less than n times) and we greedily pick the best resolution that minimizes statistical distortion for this tuple. In each iteration, we do an EMD computation using an approximation algorithm that runs in $\mathcal{O}(n^2 log^2\ n)$ time [18].

3.3 Using Closures to Compute EMD

To determine the best repair to correct an inconsistency in t_d, we must select one of a set of candidate repairs t_r. To do this, we compute the statistical distortion between $t_r \cup D_c$ and $D_{\mathcal{I}}$. We choose to use EMD to measure statistical distortion as it has been effectively used in past data cleaning and information retrieval applications [9, 26]. However, computing an exact value for EMD is still computationally expensive. The Hungarian algorithm [17] computes an exact EMD value with cubic complexity. To improve performance, many approximation algorithms have been developed in recent years (and are available in open source libraries). We use an approximation algorithm of Pele and Werman [18].

To further improve performance, Zhang et al. [26] propose optimizations to compute EMD over g groups of attributes where each group contains h non-overlapping attributes (instead of computing over all $|\mathbf{R}|$ attributes). This is favorable when h is much smaller than $|\mathbf{R}|$. Zhang et al. [26] propose to sample from the g groups and compute EMD only for selected groups rather than computing it exhaustively for all combinations of attribute groups.

We adopt a similar sampling approach to compute the EMD. However, instead of randomly dividing the $|\mathbf{R}|$ attributes into groups, we define the set of attribute groups based on the closure of attributes used in $M[t_d]$. This allows us to leverage the natural attribute relationships that exist and have already been defined by the constraints. Algorithms to compute attribute closures for FDs are well-known. In the next section, we present a new algorithm to compute attribute closures for metric FDs. We postulate that it is the correlation among these attributes that are the most important to preserve in cleaning. For each violated dependency $X \mapsto A \in M[t_d]$, we compute the closure of X,

	Organization	Latitude	Longitude	Wt
1	Univ of Toronto	43.662892	-79.395656	2/7
2	University of Toronto	43.943445	-78.895452	1/7
3	McMaster Univ	43.260879	-79.919225	1/7
4	Univ of Toronto	43.66	-79.40	1/7
5	AT&T ...	40.669550	-74.636399	1/7
6	AT&T...	40.67	-74.63	1/7

Table 3.2: Distribution of $D_{\mathcal{I}}$ over the closure Org^+.

denoted X^+, and compute EMD only over a projection on X^+ and pick maximal EMD. Hence, our computations are typically on small projections of the relation (which results in a smaller number of bins for the EMD computation) and we perform at most $|M[t_d]|$ such computations. We demonstrate the performance benefits of using attribute closures in our experimental evaluation.

EXAMPLE 3.4. *Continuing Example 3.1, without using closures, the EMD computations for the two candidate repairs t_4' and t_4'' would be computed over all six attributes of $\boldsymbol{R}$. With our optimization, we compute the closure for the antecedent of M_2 and the closure for the antecedents of M_3. In this simple example, both closures are the same: $\texttt{Org}^+ = \{\texttt{Org}, \texttt{Lat}, \texttt{Long}\}$. Hence, we would compute EMD only over the projection on these three attributes.*

For this example, let us assume that $D_{\mathcal{I}} = D_c$. This certainly does not need to be the case, but it is one option that has been proposed in the literature. The distribution of D_c and $D_{\mathcal{I}}$ is depicted in Table 3.2.

The weight vectors are shown below.

$$\begin{array}{rl} D_{\mathcal{I}} & [2/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7] \\ t_4' \cup D_c & [3/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8] \\ t_4'' \cup D_c & [2/8, 1/8, 1/8, 2/8, 1/8, 1/8, 1/8] \end{array}$$

The EMD cost for t_4' is 0.34115 which is less than for t_4'' (0.34133). These numbers reflect the intuition that t_4'' flattens what was a skewed distribution for the two possible locations for the "Univ of Toronto" (note that the two locations are consistent because of the metric FD). The repair t_4' makes the distribution more skewed but this effect is distributionally a smaller difference from the ideal than t_4''. We select the lower distortion repair and add it to D_c. This repair is set-minimal since reverting any of the attribute values in the updated t_4 would no longer satisfy M.

3.4 Pruning Low Frequency Values

When dealing with the complexity of EMD calculation, a large number of bins is proportional to a high cost in runtime of the algorithm. Therefore, without affecting the asymptotic complexity of the algorithm we attempt to reduce the number of bins used for EMD computation in order to reduce the cost. Intuitively, we want to capture the relevant portion of the distribution, that is a significant probability mass of the distribution with the minimum of bins. We implement two strategies.

First, we prune low frequency values by introducing a parameter β that captures the ratio of frequency of each bin in a distribution to the maximum bin frequency. For distributions with a long tail (e.g., Zipf), it is easy to see that the candidates whose corresponding bins are at the low end of the distribution will almost never be picked as repairs, but calculating EMD for each of those (unlikely) candidates is expensive. Computing a threshold β such that we only keep candidates with frequency greater than or equal to β allows us to optimize our algorithm. Note that each candidate has a corresponding bin in the distribution. When we prune a candidate, we also prune the bin for this candidate. So EMD computations for other candidates will be over a distribution from which low frequency bins have been removed.

	Organization	Latitude	Longitude	Frequency
1	Univ of Toronto	43.662892	-79.395656	5
2	Univ of Toronto	43.66	-79.40	4
3	Univ of Toronto	43.6628	-79.3956	1

Table 3.3: Sample distribution of data.

The parameter β lets us deal with heavily skewed distributions. At the other end of the spectrum lie distributions that are almost uniform. In that case, no matter the setting of β the number of bins (and thus iterations of EMD computations) will remain large. We use a second parameter, ϖ, to deal with large distributions where β cannot discriminate between frequencies without eliminating most of the probability mass of the distribution. Hence, when the number of bins following the pruning based on β is large, we take top-ϖ bins (according to frequency) and compute EMD.

EXAMPLE 3.5. *(using β) Consider the values in Table 3.3 with the relevant attributes sorted into bins together with a frequency count of each bin. The resulting distribution then becomes: [1/2; 2/5; 1/10]. Assume β = 30% (and ϖ = 10, therefore, the second threshold parameter does not prune tuples, since the total number of tuples in Table 3.3 is 10). The maximum frequency is 5, so that is our baseline of 100%. Since 30% (based on β) of 5 is 1.5, then all the values below the frequency of 2 are pruned. Therefore, we do not consider bin 3 as a viable repair candidate, thus reducing the number of EMD computations we need to perform.*

Setting β to 0% and ϖ to the total number of tuples in the dataset would allow the user to consider every candidate for repair and the EMD computations would be done over the full distribution (all bins). We show in our experiments that by increasing β or decreasing ϖ, we can effectively improve the efficiency of our algorithm with very little impact on the quality of the repairs that we find.

4. REASONING OVER METRIC FDS

Our repair algorithm assumes that a set of metric FDs are given as a minimal cover and makes use of attribute closures (the set of attributes metrically implied by a set of attributes). To compute minimal covers and closures, we need to understand the inference problem for metric FDs.

4.1 Metric FD Axiomatization

We present an *axiomatization* for metric FDs, analogous to Armstrong's axiomatization for FDs [2]. This provides a formal framework for reasoning about metric FDs. The axioms give insights into how metric FDs behave that are not easily seen when reasoning from first principles. Before presenting our axioms, it is interesting to note that some axioms that hold for FDs do not hold for metric FDs, including transitivity (if $X \mapsto Y$, and $Y \mapsto Z$, then $X \mapsto Z$).

EXAMPLE 4.1. *Consider this relation with two tuples.*

Person	*Position*	*Organization*	*Latitude*
RJM	*Faculty*	*University of Toronto*	*40*
RJM	*Faculty*	*Univ of Toronto*	*43.66*

The metric FD M_1: $\texttt{Person, Position} \mapsto \texttt{Organization}$ *($\theta = 6$) holds since the two organizations are within an edit*

Algorithm 2 Inference procedure for metric FDs

Input: A set M of metric FDs and a set of attributes X.
Output: The closure of X with respect to M.

1: $M_{unused} = M$; $n = 0$
2: $X^n = X$
3: **loop**
4: **if** $\exists\ V \mapsto Z \in M_{unused}$ and $V \subseteq \{X \cup W\}$, where $W = \{A \mid A \in X^n$ and $\theta_A = 0\}$ **then**
5: $X^{n+1} = X^n \cup Z$
6: $M_{unused} = M_{unused} \setminus \{V \mapsto Z\}$
7: $n = n + 1$
8: **else**
9: **return** X^n
10: **end if**
11: **end loop**
12: **end loop**

distance of 6. In addition, M_2: `Organization` $\mapsto$ `Latitude` ($\theta = .01$) *holds trivially since each tuple has a different* `Organization`. *However, the transitive dependency:* `Person, Position` $\mapsto$ `Latitude` ($\theta = .01$) *does not hold.*

THEOREM 4.2. (soundness & completeness)
These axioms are sound and complete for metric FDs.

1. Identity: $\forall X \subseteq \boldsymbol{R}$, $X \mapsto X$
2. Decomposition: *If* $X \mapsto YW$, *then* $X \mapsto Y$
3. Composition: *If* $X \mapsto Y$ *and* $Z \mapsto W$ *then* $XZ \mapsto YW$
4. Limited Reduce: *If* $XY \mapsto Z$, $X \mapsto Y$ *and* $\Theta_Y = 0$ *then* $X \mapsto Z$

Next, we define the *closure* of a set of attributes X over a set of metric FDs M. We use the notation $M \vdash X \mapsto Y$ to state that $X \mapsto Y$ is provable from M.

DEFINITION 4.3. (closure X^+)
The closure of X, denoted X^+, with respect to a set of metric FDs M, is defined as $X^+ = \{A \mid M \vdash X \mapsto A\}$.

Importantly, the closure can be used to determine if a metric FD is logically entailed from M (Lemma 4.4).

LEMMA 4.4. (closure)
$M \vdash X \mapsto Y$ iff $Y \subseteq X^+$.

4.2 Metric FD Inference Procedure

Another integrity constraint that considers similarity (difference), called differential dependencies (DDs), was introduced by Song and Chen [20]. A DD $\texttt{SequentialId}^{[0,2]} \hookrightarrow \texttt{Timestamp}^{[4,5]}$ means that if two tuples have `SequentialId` values whose difference is less than (or equal to) 2, then their `Timestamp` values must have a difference that is at least 4 and less than (or equal to) 5. By definition DDs subsume metric FDs. For instance, a metric FD `Organization` $\mapsto$ `Latitude` ($\theta = 0.01$), is equivalent to a DD $\texttt{Organization}^{[0,0]} \hookrightarrow \texttt{Latitude}^{[0,0.01]}$. Song and Chen [20] show that the inference problem for DDs is co-NP-complete. This establishes an upper bound on the complexity of inference for metric FDs. However, we show that the inference problem for metric FDs remains linear, as in traditional FDs, even though they capture additional similarity semantics.

Our inference procedure takes time proportional to the length of the dependencies in M. Our experiments show that this cost is marginal. For the 10 metric FDs described in Sec. 5, our algorithm runs in $\leq$ 1ms.

EXAMPLE 4.5. *Let $M = \{M_1, M_2, M_3\}$ be the set of metric FDs from our running example Table 1.1. To identify semantically related attributes, one can split the attributes into two groups by computing closures of the antecedents of the dependencies,* $\{\texttt{Person}, \texttt{Position}\}^+$ *and* $\texttt{Organization}^+$, *respectively. The closure* $\texttt{Organization}^+$ *is* {`Organization`, `Latitude`, `Longitude`}. *The closure* $\{\texttt{Person}, \texttt{Position}\}^+$ *is* {`Person`, `Position`, `Organization`}. *Note that for the same set of traditional FDs, the closure of* {`Person`, `Position`} *is* {`Person`, `Position`, `Organization`, `Latitude`, `Longitude`}, *as transitivity holds for FDs.*

THEOREM 4.6. (correctness of inference)
Algorithm 2 correctly computes the closure X^+.

Our proof [19] is an induction on k that uses our axioms to show that if Z is placed in X^k by Algorithm 2, then Z is in X^+. Using the completeness of our axioms, we also show that if Z is in X^+, then Z is in the set returned by Algorithm 2.

A *minimal* set of metric FDs is a set with single attributes in the consequent that contain no redundant attributes in the antecedent and that contain no redundant dependencies. We assumed that the input metric FDs for our repair algorithm in Section 3 were minimal. To achieve this, we can apply the inference procedure described above to compute a minimal cover of a set of metric FDs.

DEFINITION 4.7. (minimal cover)
A set M of metric FDs is minimal iff

1. *$\forall\ X \mapsto Y \in M$, Y contains a single attribute;*
2. *for no $X \mapsto A$ and proper subset Z of X is* $\mathrm{M} \setminus \{X \mapsto A\} \cup \{Z \mapsto A\}$ *equivalent to* M;
3. *for no $X \mapsto Y \in M$ is $M \setminus \{X \mapsto A\}$ equivalent to M.*

If M is minimal and M is equivalent to a set of metric FDs N, then we say M is a minimal cover of N.

THEOREM 4.8. (minimality)
Every set of metric FDs M has a minimal cover.

From the soundness of our axioms, it is possible for every set of metric FDs, to create an equivalent set with only a single attribute on the right-hand side. For a metric FD $X \mapsto A$, the second condition can be satisfied by checking for each $B \in X$ if $A \in \{X \setminus B\}^+$. For the third condition, we can test whether $X \mapsto A$ is redundant by computing closure X^+ with respect to $M \setminus \{X \mapsto A\}$.

5. EXPERIMENTAL STUDY

We present an experimental evaluation of our techniques. Our evaluation focuses on four objectives.

- An evaluation of the effectiveness of our approach using real and synthetic datasets (Section 5.3).
- A comparative study of our algorithm with other approaches, quantifying the benefits of our approach in terms of statistical distortion and accuracy (Sec. 5.3.3).
- Scalability and performance (Section 5.4).
- Robustness over a number of input parameters and different problem characteristics (Section 5.4).

5.1 Setup

Our experiments were performed on an Intel Xeon X3470 machine with 8 2.93GHz processors and 23GB of RAM on Ubuntu 12.04. All algorithms were implemented in Java.

Sym.	Description	Values
N	# of tuples	100K, 500K, **1M**, 2M, 3M
γ	Zipf distribution	0.01, 0.25, 0.5, **0.75**, 0.99
e	error percentage	3%, **5%**, 7%, 10%
F	# of FDs	1-10, default: **3**
β	candidate pruning	0, 30, **50**, 70%
ϖ	distribution pruning	20, **50**, 100, 500

Table 5.1: Parameters and defaults (bolded).

Table 5.1 summarizes the main parameters that we used, the range of values we considered, and the default value for the parameter (in bold). The default value is used unless otherwise mentioned. We used the UIS Database generator [22] to generate datasets with a Zipf distribution containing N tuples (N varied from 100K to 3M). The error introduced is varied from 3-10%. An error rate of 3% for a dataset D means there is a maximally clean subset D_c such that $|D - D_c|/|D| = 0.03$.

In addition to synthetic data, we used two real datasets. The Flight dataset [11] containing 26,987 flights and for which ground truth is provided. We also used the CORA dataset, a bibliographic dataset with 1300 tuples, for which ground truth has also been provided [8]. We report on the precision of our algorithm, computed using the ground truth (Section 5.3.3).

We ran each experiment six times and report the average and maximum SD over these runs (each using a different random order of tuples). We perform cleaning w.r.t. different numbers of metric FDs F (from 1 to 10). For all experiments, we compute EMD over attribute closures (Section 3.3) rather than over the full set of attributes $\mathbf{R}$. In addition, to improve performance, our algorithm takes two pruning parameters as input β (for pruning candidates and their corresponding bins with relative frequencies under β) and ϖ (pruning all but the ϖ most frequent bins). When $\beta = 0\%$ and $\varpi = N$ (or $\varpi \geq$ maximum number of bins in a distribution) we perform no pruning. Our experiments vary β from $0 - 70\%$ and ϖ from $20 - 500$.

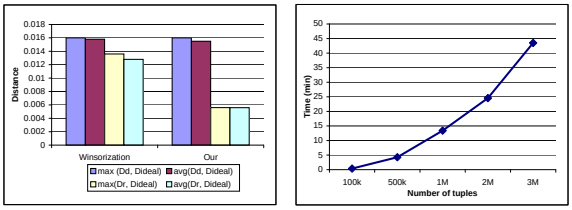
5.2 Data Creation

Clean Data The UIS generator can create clean data that satisfies the ten FDs in Figure 5.1. For our experiments, we needed metric FDs and data conforming to these metric FDs. To do this, we set $\theta_{SSN} = 0$, leaving constraint 8 as a traditional FD. For the string attributes (`FirstName`, `LastName`, `StreetAddr`, `State`, `City`), we used Jaro-Winkler as the similarity metric and set $\theta = 0.75$. Thus, FDs 1, 2, 4, 6, 7, 9, and 10 are turned into metric FDs using this metric and threshold. For the numeric attributes (`StreetNum` and `Zip`), we use absolute difference as the metric and a threshold $\theta = 3$. We then perturb the data (which satisfies the FDs) to create a DB that satisfies the metric FDs (but not necessarily the original FDs). We limit our changes to 10% of the data and denote the result as D_{gold}.

Dirty Data We modified D_{gold} to create a dirty relation by injecting two kinds of violations. *Consequent violations*: for two tuples t_1 and t_2 that satisfy some metric FD $X \mapsto A$ we modify $t_1[A]$ to a value that is picked uniformly from the distribution of A, such that the $m_A(t_1[A], t_2[A]) > \theta_A$. *Antecedent violations*: for three tuples t_1, t_2 and t_3 where $t_1[X] = t_2[X]$, t_1 and t_2 satisfy some metric FD $X \mapsto A$, $m_A(t_1[A], t_3[A]) > \theta_A$ and $t_1[X] \neq t_3[X]$, we modify $t_1[X]$, such that $t_1[X] = t_3[X]$. We refer to the resulting instance as the dirty dataset D_d.

1) {`SSN`} ↦ {`FirstName`}; 2) {`SSN`} ↦ {`LastName`}
3) {`SSN`} ↦ {`StreetNum`}; 4) {`SSN`} ↦ {`StreetAddr`}
5) {`SSN`} ↦ {`ZIP`}; 6) {`Zip`} ↦ {`City`}; 7) {`Zip`} ↦ {`State`}
8) {`FirstName`, `LastName`, `MidInitial`} ↦ {`SSN`}
9) {`FirstName`, `LastName`, `MidInitial`} ↦ {`City`}
10) {`FirstName`, `LastName`, `MidInitial`} ↦ {`State`}

Figure 5.1: Metric FDs over UIS Dataset.



(a) Vs Winsorization. (b) Num tuples vs time.

Figure 5.2: Distance of repair & scalability.

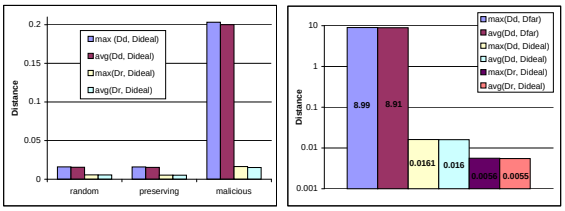
5.3 Repair Quality

To assess the accuracy of our repairs, we begin with a study where we compute a maximally clean subset of D_d, D_c, and set $D_{\mathcal{I}} = D_c$.

5.3.1 Accuracy Study

First, to understand if our (heuristic) approach to minimizing statistical distortion has a significant influence on the distortion of repairs, we compared it against another repair strategy, called *Winsorization* that for an error e picks the closest acceptable value to e as its repair. Winsorization was also used for comparison by Dasu and Loh [9] and would be a natural choice for a logical repair technique that minimizes the cost (measured by distance) of a change [7]. This approach does not attempt to preserve the distribution and thus, we expect our repair algorithm to do better at minimizing statistical distortion. Figure 5.2a shows the distortion (max and average) of a set-minimal repair produced by Winsorization compared to our approach confirming that we are able to find significantly better (lower distortion) repairs. On average, in our experiments consequent-repairs were chosen approximately 80% of the time compared to 20% of the time for antecedent-repairs. Our algorithm selected to repair both the consequent and antecedent of a tuple only in a very small number of cases.

While the above experiment illustrates that our algorithm significantly reduces the distortion compared to another natural method, it does not tell us how good our approximation is. For this purpose, we designed the following experiment. We generate dataset D_{silver} from dataset D_{gold} by replacing values in D_{gold} one at a time by corresponding attribute values in D_d as long as this modification does not cause a violation. That is, with D_{silver} we approximate the repair we should get from D_{gold} given our condition of set-minimality (no unnecessary changes). Only for this particular experiment, we use D_{gold} as $D_{\mathcal{I}}$ (i.e., $D_{\mathcal{I}} = D_{gold}$), because we attempt to show how close to the original ground truth dataset we can get in the best possible scenario, given the requirement that the final repair has to be set-minimal. (Therefore, we may not be able obtain D_{gold} itself as a repair.) The relation D_{silver} can then be considered a lower bound on statistical distortion from D_{gold}. Hence, EMD(D_{silver}, D_{gold}) is the best we can achieve, and our objective is to verify how close our repair comes to this



(a) Varying error injection. (b) Distance wrt D_{far}.

Figure 5.3: Distance in context.

"best" distance. We expect EMD(D_d, D_{gold}) >> EMD(D_r, D_{gold}) > EMD(D_{silver}, D_{gold}). This expectation is borne out by experimental data where the best possible distance of 0.0038 is very close to the distance of our repair to D_{gold} 0.0041 (compared to a distance of 0.162 for dirty data).

Next, we show that our algorithm achieves low statistical distortion over different $D_{\mathcal{I}}$'s. We varied the ideal dataset by changing 1% to 3% of the tuples in $D_{\mathcal{I}}$ with the goal of showing the stability of our approach. In more detail, we vary the ideal dataset by randomly updating values of attributes in $D_{\mathcal{I}}$ so that the distribution of attributes is different. In this experiment, we observe that the change of $D_{\mathcal{I}}$ had only a very slight impact on the returned statistical distortion of D_r. The distance varies only in the 4th or higher decimal place (e.g. 0.005655 to 0.005654).

5.3.2 Distribution Preserving Data

To further stress test the ability of our algorithm to consistently produce a low statistical-distortion repair we took the original dataset D_{gold} and injected errors following a specific distribution, rather than randomly. We applied two approaches (a) *preserve* D_{pres}: the distribution of the errors mimics the distribution of the original dataset, (b) malicious D_{dest}: the distribution of the errors *destroys* the distribution of the original dataset. First, we calculated the histogram of each attribute and then picking the new values to perturb the cells according to this distribution, either intentionally preserving it or intentionally changing the distribution. In this way, we preserve the marginal distributions, or alternatively change the distribution of the dataset by purposefully shifting the values into a single quantile of the original distribution.

We calculate the statistical distortion from our repair to $D_{\mathcal{I}}$ and compare it to the EMD(D_{pres}, $D_{\mathcal{I}}$) and EMD(D_{dest}, $D_{\mathcal{I}}$). In Figure 5.3a we observe that the distance for the dataset with malicious error injection (0.0164) is much further from $D_{\mathcal{I}}$ than either data preserving or random error injection dataset (0.0053 and 0.0056 respectively), as expected, and that in the worst case (with a distribution destroying errors which are not independent or random) we still successfully minimize statistical distortion. The relation D_{pres} is slightly closer in terms of statistical distance (for both D_d and D_r) than the datasets with randomly injected errors. However, it is noteworthy that the difference between EMD(D_r, $D_{\mathcal{I}}$) is larger for the D_{dest} dataset than for D_{pres}. Overall, our conclusion is that the quality of the repair is affected by the error injection method; however, this effect is small and we are still able to achieve small distortion.

Next, we use DBGen to generate a synthetic dataset D_{far}, which is statistically far from $D_{\mathcal{I}}$. This can be accomplished by using correlations between the multiple dimensions (attributes) to model the effect of skewed data. While in the case of the data D_{dest} we inject 5% error by shifting the distribution, in this case we shift the entire dataset. Let δ = EMD($D_{far}, D_{\mathcal{I}}$). Our hypothesis is that all of our repairs will fall between $[0, \delta]$, and will lie much closer to zero than δ. In Figure 5.3b, D_{far} has a distance of 8.99 compared to 0.005653 of our repair, meaning that our repair is statistically very close to $D_{\mathcal{I}}$ (and much closer than D_d).

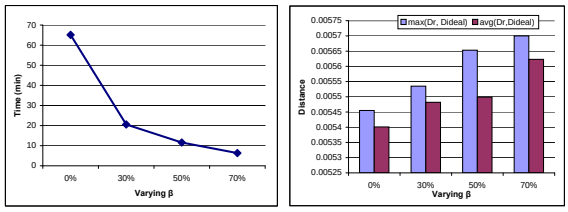
5.3.3 Comparative Study

We compare our algorithm against another algorithm, the Unified Repair Model by Chiang and Miller [8], that performs data repairs as well as constraint repairs, but for FDs, not metric FDs. Their algorithm recommends data and FD repairs. To ensure a fair comparison, we ran the algorithm with only the data repair option. We ran this experiment on the real bibliographic dataset CORA (which has only 1300 tuples and two FDs. (For detailed description of the CORA dataset see original paper [8].) The two FDs we consider are {`Title`, `Venue` } → `Authors` and {`Venue`, `Year`} → `Location`. For this experiment, we computed the repairs using both algorithms and then checked the statistical distortion compared to the dirty dataset for our repair (D_r) and the repair produced by the Unified Repair Model (D_{um}). For this experiment, $EMD(D_d, D_{\mathcal{I}})$ is 0.107. Our approach produces a D_r with lower distortion $EMD(D_r, D_{\mathcal{I}}) = 0.105$ compared to the Unified Model $EMD(D_{um}, D_{\mathcal{I}}) = 0.114$. Because the Unified Model algorithm is cost based, it repairs errors to high frequency values possibly leading to skewed distributions. As a result, certain values are over-represented in the repaired solution. Our solution keeps the distribution more consistent with that of the ideal distribution. The precision of our algorithm on this dataset was 86.4% compared to the results from the Unified Repair Model of 83.9%. We calculated precision as: $precision = (\#CorrectRepairs)$ / $(\#TotalRepairs)$, where $\#CorrectRepairs$ is the number of correct repairs checked manually by verifying the correctness of the data on the web and $(\#TotalRepairs)$ is the total number of performed repairs. The results of this experiment show that our algorithm can achieve comparable precision to the Unified Model and still achieve significant gains in minimizing the distortion of the repair. This experiment illustrates the benefits of using a combined statistical and logical approach, since our algorithm achieves a comparable precision and a lower statistical distortion given the same set of data.

We also performed experiments on the real flights data [11], an integration of several online data sources. This dataset contains 26987 tuples for a single day of flights (2011-12-01), and two metric FDs: `FlightId` ↦ `DepartureTime` and `FlightId` ↦ `ArrivalTime`. We only present results for one day out of thirty since repairing each day's worth of data is independent of other days. For this experiment, we have a set of flights whose information has been manually verified by using a trusted source (airline website), this is the ground truth dataset. We computed the precision with Unified Model (83.2%) and with our algorithm (82.0%).

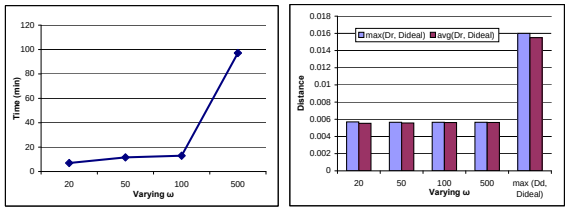
5.4 Scalability and Performance

We measure performance by computing the system time for running the repair algorithm over the synthetic datasets. We also demonstrate the benefits (and impact on accuracy) of our pruning strategies.



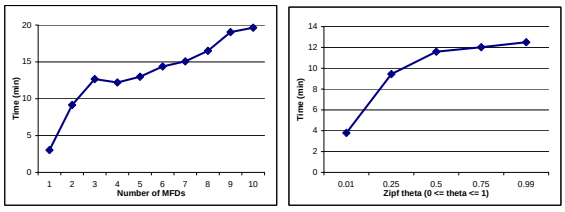
(a) Vs time (min). (b) Vs EMD distance.

Figure 5.4: Effect of lower number of iterations.



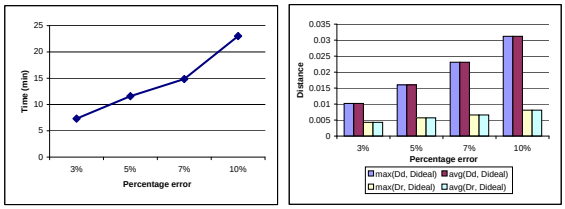
(a) Vs time (min). (b) Vs EMD distance.

Figure 5.5: Effect of pruning lower frequency candidates.



(a) Number of constraints. (b) Varying the Zipf.

Figure 5.6: Varying parameters vs time.



(a) Vs time (min). (b) Vs EMD distance.

Figure 5.7: Percentage error.

5.4.1 *Impact of Pruning*

As described, in Section 3.4, we proposed two pruning strategies (based on parameter β and ϖ) to make our algorithm more efficient. In Figure 5.4, we vary the parameter β and measure its effect on statistical distortion and on the repair algorithm runtime. While increasing β, the accuracy decreases moderately, β= 0% (i.e., no pruning of repair candidates) producing the lowest statistical distortion. Meanwhile, the runtime decreases. Since the accuracy only decreased a small amount from β=30% to β=50% (0.0056 to 0.0054, compared to the distance of 0.016 for the D_d to $D_{\mathcal{I}}$) we set the default β to 50% for other experiments (including the previously described experiments). Note that the runtime for $\beta = 50\%$ is 12 min. compared to 65 min. for $\beta = 0\%$, with comparable distance (0.005653 for $\beta = 50\%$ vs. 0.005455 for $\beta = 0\%$).

We also measure the effects of pruning the number of bins used in the EMD calculations, by varying the parameter ϖ that, for large distributions, controls how many bins we keep for the calculation. Note that setting a cut-off of 100 bins over 500 bins has negligible effect on accuracy (e.g., distance of 0.005655 vs 0.005653) and great improvement in performance (approximately 12 min. vs 97 min.) as shown in Figure 5.5. Note also that the repair algorithm fails to finish after 24 hours if we do not use either β or ϖ parameters, so they are necessary for effective calculation.

5.4.2 *Number of Tuples and Constraints*

In Figure 5.2b, we vary the number of tuples. Our runtimes are roughly 12 minutes for 1M tuples, and roughly 43 minutes for 3M tuples. The runtime is partly dependent on the library we used to calculate EMD which is a super-quadratic (in the number of bins) approximation on the exact cubic algorithm [17]. These results compare favorably to other approaches, such as the Unified Model [8].

Furthermore, we study the impact of varying the number of constraints for N = 1M, e = 5% and $\mu \in [2, 4]$ on our synthetic dataset. Figure 5.6 shows that the repair accuracy scales well as the number of constraints increase, despite the increase in search space. We select ten of the metric FDs that hold over our synthetic relation, and randomly remove one metric FD at a time until we are left with one. We report the average time, with the default 3 metric FDs yielding roughly 12 min. runtime.

5.4.3 *Varying the Distribution and Error Rate*

We varied the parameter that controls the Zipf distribution while generating the synthetic data, $0 \leq \gamma \leq 1$ where 1 is uniform distribution. We ran experiments for $\gamma = \{0.01, 0.25, 0.5, 0.75, 0.99\}$. Figure 5.6 shows the time increasing as the distribution becomes more uniform (and the number of iterations required to find the repair candidate increases). By default, we set $\gamma = 0.75$. We also experimented with various sizes of attributes in a matrix that is used to compute EMD. Our findings are that the number of attributes impacts the calculation significantly, however, not as much as the active domain of each attribute. If we have many attributes but the domain only contains two values (e.g., "Male" and "Female") then the EMD computation is fast. However, if the active domain for the same closure is big, the EMD computation will be significantly slowed by the number of possibilities that have to be considered.

Figure 5.7 shows the accuracy and scalability as the error rate e increases. We fix N = 1M tuples, with $\mu \in [2, 4]$. The accuracy decreases moderately with increasing error, from 0.0043 for 3% to 0.0081 for 10% (both smaller than $max(D_d, D_{\mathcal{I}})$ of 0.0102 and 0.0312 respectively).

6. RELATED WORK

Our work combines logical and quantitative data cleaning. We have made contributions that relate to each of these fields in addition to proposing a novel problem definition and solution that combines the benefits of both.

Our approach uses metric FDs [16], one of many logical formalisms that permit the expression of constraints that use similarity rather than exact equality. While metric FDs permit similarity only in the consequent, one could certainly define an alternate formulation with similarity in the antecedent. With a similar motivation, Fan et al. [10] proposed *matching dependencies* (MDs). In contrast to metric FDs, MDs have been used in data cleaning. MDs can be

defined across multiple relations and permit similarity operators over the antecedent. Fan et al. [10] present a quadratic inference procedure for MDs and a complex axiomatization for MDs (with 11 axioms). Notably, ours is the first work to consider how to use metric FDs in data cleaning and the first to present an axiomatization and inference system for metric FDs. Inference for metric FDs is linear (and characterized by 4 axioms) in contrast to MDs. An interesting extension of our work would be to consider the problem of statistical-distortion minimal repairs for MDs.

Song and Chen [20] introduced *differential dependencies* (DDs) which specify that when two tuples have antecedent values within a specified range (specified by a min and max value), then their consequents must be within a specified consequent range. A related notion is that of *sequential dependencies* (SDs), which specify that when tuples have consecutive antecedent values, their consequents must be with a specified range [13]. The verification problem has been studied for approximate SDs [13]. To the best of our knowledge, inference over SDs has not been studied. For DDs, the inference problem is co-NP-complete (and has been used to define minimal covers for DDs) [20].

Existing logical cleaning approaches strive to find repairs that are minimal with respect to the number or cost of modifications without using any statistical properties to guide the repairs. While some quantitative notions of logical repairs exist for denial and aggregate constraints [4] that rely on finding the closest (lowest cost) repair, these repairs can dramatically change the statistical properties of the data [9]. In contrast, quantitative data cleaning uses statistical properties to guide the selection of repairs [3, 9, 14]. Yakout et al. [25] present a repair measure based on the likelihood benefit in approaching an ideal distribution. None of these quantitative cleaning approaches guarantee that a repair will satisfy a set of integrity constraints.

We used the EMD approximation algorithm of Pele and Werman [18] in our experiments. More recently, Tang et al. [21] provide a new optimization to what they call the refinement phase of EMD, making EMD computation even more scalable. This work is complementary to our work as our optimizations are restricted to the filtering phase of EMD. Hence, Tang et al.'s innovations fully apply to our algorithm and could be used in future work to further improve the scalability of our approach.

7. CONCLUSIONS AND FUTURE WORK

We believe ours is the first approach to combine the benefits of logical data cleaning (that effectively uses data quality rules to specify both what is an error and what is a correct repair) with quantitative data cleaning (that uses distributional properties of data to find a good repair that preserves these properties). We provided a first instantiation of this approach that uses metric FDs to specify when data is erroneous, and that uses EMD to measure the statistical distortion between a repair and a desired distribution.

Our approach can certainly be applied to other measures of statistical distortion and to other classes of constraints. We believe this is an important direction for data cleaning – combining the successes of both logical and statistical approaches to provide more robust cleaning solutions.

8. REFERENCES

[1] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79, 1999.

[2] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.

[3] L. Berti-Equille, T. Dasu, and D. Srivastava. Discovery of complex glitch patterns: A novel approach to quantitative data cleaning. In *ICDE*, pages 733–744, 2011.

[4] L. Bertossi, L. Bravo, E. Franconi, and A. Lopatenko. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. *Information Systems*, 33(4-5):407–434, 2008.

[5] G. Beskales, I. F. Ilyas, and L. Golab. Sampling the repairs of functional dependency violations under hard constraints. *PVLDB*, 3(1):197–207, 2010.

[6] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755, 2007.

[7] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *SIGMOD*, pages 143–154, 2005.

[8] F. Chiang and R. J. Miller. A unified model for data and constraint repair. In *ICDE*, pages 446–457, 2011.

[9] T. Dasu and J. M. Loh. Statistical distortion: Consequences of data cleaning. *PVLDB*, 5(11):1674–1683, 2012.

[10] W. Fan, X. Jia, J. Li, and S. Ma. Reasoning about record matching rules. *PVLDB*, 2(1):407–418, 2009.

[11] Flights data. `http://www.lunadong.com/fusionDataSets.htm`.

[12] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C. Saita. Declarative data cleaning: Language, model, and algorithms. In *VLDB*, pages 371–380, 2001.

[13] L. Golab, H. Karloff, F.Korn, A. Saha, and D. Srivastava. Sequential dependencies. *PVLDB*, 2(1):574–585, 2009.

[14] J. Hellerstein. Quantitative data cleaning for large databases. In *Technical report, UC Berkeley*, Feb 2008.

[15] S. Kolahi and L. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, pages 53–62, 2009.

[16] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric Functional Dependencies. In *ICDE*, pages 1291–1294, 2009.

[17] O. Pele and M. Werman. A linear time histogram metric for improved SIFT matching. In *Eur. Conf. on Computer Vision*, pages 495–508, 2008.

[18] O. Pele and M. Werman. Fast and robust earth mover's distances. In *IEEE Int. Conf. on Computer Vision*, pages 460–467, 2009.

[19] N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. Combining quantitative and logical data cleaning. April 2015. `http://dblab.cs.toronto.edu/project/DataQuality`.

[20] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *TODS*, 36(3):16, 2011.

[21] Y. Tang, L. H. U, Y. Cai, N. Mamoulis, and R. Cheng. Earth mover's distance based similarity search at scale. *PVLDB*, 7(4):313–324, 2013.

[22] UIS Data Generator. `http://www.cs.utexas.edu/users/ml/riddle/data.html`.

[23] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. In *ICDE*, pages 244–255, 2014.

[24] X. Wang, X. L. Dong, and A. Meliou. Data x-ray: A diagnostic tool for data errors. In *SIGMOD*, pages 1231–1245, 2015.

[25] M. Yakout, L. Berti-Equille, and A. K. Elmagarmid. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *SIGMOD*, pages 553–564, 2013.

[26] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. *PVLDB*, 3(1):805–814, 2010.