Information Sciences 507 (2020) 386–403

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Multi-source data repairing powered by integrity constraints and source reliability

Chen Ye[a], Hongzhi Wang[a,*], Kangjie Zheng[a], Jing Gao[b], Jianzhong Li[a]

[a] Harbin Institute of Technology, Harbin, China
[b] SUNY Buffalo, Buffalo, USA

ARTICLE INFO

Article history:
Received 12 December 2018
Revised 16 August 2019
Accepted 18 August 2019
Available online 22 August 2019

Keywords:
Data repairing
Conflict resolution
Denial constraints

ABSTRACT

It is crucial to identify and resolve the inconsistencies and conflicts in data. To tackle the inconsistencies, integrity constraints are involved to constrain the attribute values of related entities. As for the multi-source conflicts, the true values of each entity are identified by trusting the reliable sources. In practice, it is common that inconsistencies and conflicts simultaneously appear. To deal with this case, traditional techniques would separately resolve the inconsistencies and conflicts, by conducting different approaches based on the above principles. However, such a procedure may not be the appropriate solution. Specifically, locally resolving conflicts for a certain entity may overlook the information from its related entities, while enforcing constraints on related entities may miss correct values of these entities in turn. To jointly resolve the inconsistencies and conflicts, this paper proposes a novel technique powered by integrity constraints and source reliability. The key component of our solution is to incorporate denial constraints, an expressive type of integrity constraint, into the process of conflict resolution. We formulate it as an optimization problem and develop an iterative algorithm to solve it. Benefiting from this algorithm, the repaired result is not only supported by reliable sources but also satisfies the denial constraints. Additionally, we also propose two optimal strategies to ensure that it is scalable under massive constraints. Experimental results on real-world datasets demonstrate the high accuracy and scalability of the proposed approach.

1. Introduction

Data quality improvement approaches have been studied for decades, among which inconsistencies and conflicts are the two main issues to deal with. In a database, *inconsistencies* may occur among attribute values within a single entity or between different pairwise entities [12]. While in multiple databases (sources), *conflicts* may occur when multi-source attribute values are provided for the same entity [43]. To resolve multi-entity inconsistencies and multi-source conflicts, constraint-based data repairing and conflict resolution play important roles.

Constraint-based data repairing is the process of enforcing integrity constraints or rules on the attribute values of a set of entities [12]. The inconsistencies w.r.t. the constraints are then considered to be in violation. Different principles [4,9,14,21,32,35] have been proposed to repair these violations so that the repaired result satisfies the constraints.

* Corresponding author.
E-mail addresses: yech@hit.edu.cn (C. Ye), wangzh@hit.edu.cn (H. Wang), kangjie.zheng@gmail.com (K. Zheng), jing@buffalo.edu (J. Gao), lijzh@hit.edu.cn (J. Li).

https://doi.org/10.1016/j.ins.2019.08.044
0020-0255/

Table 1
Tax information tables.

	$\mathcal{X}_1$				$\mathcal{X}_2$				$\mathcal{X}_3$			
Entity	zip	city	salary	tax	zip	city	salary	tax	zip	city	salary	tax
Bob	10002	NYC	60000	5000	10003	NYC	60000	5000	10002	LA	60000	4950
Kate	-	-	-	-	10002	BUF	35000	3500	10002	NYC	30000	3000
Mike	14221	BUF	20000	30000	14221	BUF	23000	2000	14221	NYC	30000	3800

Table 2
Ground truth and data repairing results.

	Ground Truth				CRH				HoloClean			
Entity	zip	city	salary	tax	zip	city	salary	tax	zip	city	salary	tax
Bob	10002	NYC	60000	5000	10002	NYC	60000	5000	10002	NYC	60000	5000
Kate	10002	NYC	35000	3500	10002	**BUF**	35000	3500	**10003**	NYC	35000	3500
Mike	14221	BUF	24000	3000	14221	BUF	**20109**	**28983**	**10002**	**NYC**	**23000**	**2000**

Conflict resolution (i.e., truth discovery) aims to aggregate accurate information from multi-source data [24,25]. Traditional work resolves conflicts typically by taking, e.g., the max, min, avg, any of attribute values, while state-of-the-art approaches [10,22,33,43,45,46] usually take the source reliability into consideration. That is, the aggregated result is obtained by trusting the information provided by more reliable sources.

When the inconsistencies and conflicts simultaneously occur, a natural way is to separately conduct constraint-based data repairing and conflict resolution methods. However, such a procedure may not be the best solution. Locally resolving conflicts for a certain entity may overlook the information from its reliable entities, while enforcing constraints on related entities may miss correct values of these entities in turn. We use an example to illustrate the drawbacks of separately conducting two processes, and then motivate our approach.

Example 1. Consider three sources (databases) that provide the tax information of a group of people. Table 1 shows the information tables $\mathcal{X}_1$, $\mathcal{X}_2$, and $\mathcal{X}_3$, which store all the information provided by sources S_1, S_2, and S_3, respectively. We use "-" to represent that the source does not provide any information for a specific person. Table 2 shows the ground truth and data repairing results. The errors are highlighted in bold.

First, we focus on data repairing methods that rely on integrity constraints. Suppose that the following constraints apply for the attribute values of the entities: (1) two persons with the same zip code live in the same city; (2) it is impossible for one person to have his tax greater than his salary;

(3) two records referring to the same entity have the same zip, city, salary, and tax. Note that constraint (3) is used to resolve the conflicts among multiple sources. A repaired result using a state-of-the-art method HoloClean [32] is shown in Table 2. Take Kate as an example, with constraint (1), HoloClean detects a violation among Bob's zip, Bob's city, Kate's zip, and Kate's city in $\mathcal{X}_3$. These cells are then marked as dirty cells which need to be repaired. Considering the repaired value for Kate's zip, HoloClean takes the values which co-occur with the value of Kate's city (i.e., "NYC") in the data as candidate repaired values (i.e., "10002" from Bob's zip in $\mathcal{X}_1$; "10003" from Bob's zip in $\mathcal{X}_2$; "14221" from Mike's zip in $\mathcal{X}_3$). Based on the probability inference, HoloClean then repairs the value of Kate's zip to "10003". Similar repairing processes are performed for all the dirty cells under all the constraints. We can see that the repaired result of HoloClean satisfies the given constraints. However, without extra information about the correct values, e.g., the reliability information of sources, it is difficult to resolve the inconsistencies with the correct values. As a result, the data repairing approach tends to obtain a consistent result without a guaranteed accuracy.

We then focus on conflict resolution methods that aim to estimate source reliability. Suppose that S_1 is more reliable than S_2, and S_2 is more reliable than S_3. The truths should be closer to the information provided by S_1. Considering the result of the widely adopted method CRH [23], compared to the ground truth, CRH achieves a high accuracy in most cases by correctly identifying S_1 as reliable. However, when S_1 provides inaccurate values, e.g., the tax of Mike claimed by S_1 is "30000" whereas the ground truth is "3000", CRH will achieve the biased result "28983" by calculating the truth closer to "30000" compared with "2000" provided by S_2 and "3800" provided by S_3. Thus, without extra information, identifying the correct values only by a source reliability estimation is also not reliable.

From the above example, we can infer that the information falls short for both processes by separately considering the inconsistencies and conflicts. If we combine the information obtained in different processes, we can repair more errors in multi-source data, i.e., identifying the correct values which are both consistent and reliable. In this paper, we introduce a novel data repairing approach that simultaneously resolves inconsistencies and conflicts. Instead of considering each aspect in isolation, we use all available information, i.e., integrity constraints and source reliability degrees, to suggest data repairs. On the one hand, constraints are helpful to identify the errors made by reliable sources. On the other hand, multi-source information gives extra evidence to resolve the violations.

Challenges. Data repairing jointly for inconsistencies and conflicts complements the evidence of both processes. However, this idea also brings several challenges. (1) Which kind of constraint is beneficial and available for multi-source data? (2) How to get the repaired result accurately from multi-source conflicting data under such constraints? (3) How to ensure the scalability when the number of constraints is relatively large?

In the following sections, we address these challenges. For the first challenge, we adopt denial constraints (DCs) [6,13], a universally quantified first-order logic formalism which can express a large number of effective and widely existing constraints, such as check constraints [20], functional dependencies (FDs) [4], and conditional functional dependencies (CFDs) [11]. Thus, various types of relationships among entities can be expressed through DCs. Additionally, DCs are easily achieved through consulting with domain experts and conducting existing DC discovery approaches [2,7].

For the second challenge, we formulate the data repairing problem as an optimization problem, where the source reliability degrees and the repaired result are defined as two sets of unknown variables, and DCs are defined as constraints. The objective is to minimize the overall weighted deviation between the repaired result and multi-source information under the DCs, where each source is weighted by its reliability degree. We then propose an iterative algorithm to efficiently solve the optimization problem.

For the third challenge, two optimal strategies are proposed to ensure the scalability of the proposed algorithm.

Contributions. We summarize our contributions as follows.

- In order to obtain an accurate repaired result, we adopt DCs for their wide existence and easy discovery. To the best of our knowledge, this is the first study of the data repairing problem for jointly resolving inconsistencies and conflicts.
- We formulate the data repairing problem as an optimization problem and propose solutions by conducting an iterative procedure to jointly infer the source reliability degrees and the repaired result under the DCs.
- To reduce the total execution time, we propose two scalable strategies for the iterative procedure by optimal grouping entities and pruning candidate values.
- We conduct extensive experiments with four real-world datasets. The experimental results clearly demonstrate that the proposed method outperforms both the constraint-based data repairing and conflict resolution baselines.

Organization. Related work is discussed in Section 2. We introduce several concepts and define the data repairing problem in Section 3. In Section 4, we propose an iterative algorithm and scalable strategies to solve the problem. We conduct extensive experiments on four real-world datasets, and validate the effectiveness and efficiency of the proposed method in Section 5. Finally, we summarize the paper in Section 6.

2. Related work

Improving data quality is one of the most important issues in databases, and has been studied for years [1,4,9,10,14,17,22,23,35,39,41,43,47–50]. In this section, we discuss two directions of work, namely, constraint-based data repairing and conflict resolution (i.e., truth discovery), that are related to this paper.

Constraint-based Data Repairing.

Constraint-based data repairing is a useful tool to resolve inconsistencies within a database [11]. The main idea is to use data dependencies to capture the semantic errors, and explicitly define a principle to correct these errors. To detect and repair these errors, various data repairing methods have been proposed based on functional dependencies (FDs) [4,21], functional dependencies (FDs) and inclusion dependencies (INDs) [4], conditional functional dependencies (CFDs) [11], conditional inclusion dependencies (CINDs) [14], denial constraints (DCs) [8,32], editing rules [17], fixing rules [35], etc. As data dependencies on their own are insufficient to correctly resolve the violations, master data [17] and confidence values placed by users [4,9,16] are used to guide the process of repairing. To ensure the accuracy of the generated repairs, the help of user confirmation is needed [17,27,39]. In contrast to these methods, our method does not need any prior knowledge or supervision. Taking advantage of the information of DCs and source reliability degrees, our method is able to find a repaired result not only consistent but also achieves a high accuracy, which cannot be handled by the existing data repairing methods.

Conflict Resolution.

Conflict resolution aims to combine data from multiple sources into a single representation [3]. Traditional methods resolve conflicts typically by selecting the max, min, avg, any value [15]. This work differs from the traditional ones by taking the source reliability degrees into consideration, and do not assume the availability of timestamps.

There has been work on truth discovery [24,25,43], which tries to infer the true facts as well as the source reliability degrees from the data without any supervision. Various scenarios have been considered in truth discovery, such as dependent data sources [10,44], correlation between entities [26], continuous and heterogeneous data types [23,45], long-tailed data [22,37], hardness of obtaining the correct data [18,33], multiple truths problem [31,36,46], and incorporating prior knowledge [29,30]. Among these scenarios, as far as we know, the only direction of work similar to ours is [29,30] proposed by Pasternack et al. However, their work focuses on incorporating common-sense knowledge which often does not hold true. Thus, the accuracy of these methods could not be guaranteed. In contrast, our approach focuses on incorporating DCs [8], which are widely used to capture the inconsistencies and conflicts among the real-world entities. The problem is formulated as an optimization framework, and the proposed solutions have theoretical guarantees. Moreover, with the help of DCs, the proposed approach is able to repair the errors among related entities in different data types (e.g., categorical data and continuous data), which cannot be handled by the above methods.

Table 3
Table of notations.

Symbol	Definition
L	The number of entities
P	The number of attributes
K	The number of sources
v_{lp}^k	The value of the pth attribute for the lth entity provided by the kth source
$\mathcal{X}_k$	The information table which contains the values of P attributes for L entities provided by the kth source
w_k	The reliability score of the kth source
$\mathcal{W}$	The set of reliability scores towards K sources
v_{lp}^*	The repaired value of the pth attribute for the lth entity
$\mathcal{X}^*$	The repaired table which contains all repaired values of P attributes for L entities
Σ	The set of DCs defined on $\mathcal{X}^*$
φ	A DC from Σ
B	The set of operators defined on φ
$\odot$	An operator from B
C_z	A clause of φ
$r(v_{im}^*)$	A constant c_{im}^* or a variable v_{jn}^* which made up of C_z with v_{im}^*
$v_{im}^* \odot r(v_{im}^*)$	The form of C_z

3. Problem definition

In this section, we define the problem of data repairing for inconsistencies and conflicts. We first introduce some concepts adopted in this paper. The problem is then defined. Table 3 summarizes the notations used in this paper.

Suppose L entities are provided by K sources, and each entity is made up of P attributes. The value of the pth attribute for the lth entity provided by the kth source is denoted as v_{lp}^k. The information table provided by the kth source is denoted as $\mathcal{X}_k$, where v_{lp}^k is its lpth entry. The repaired value of the pth attribute of the lth entity is denoted as v_{lp}^*. Given K information tables $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k\}$, the repaired result of all the entities on all the attributes are stored in a repaired table $\mathcal{X}^*$, whose lpth entry is v_{lp}^*. Source weights are denoted as $\mathcal{W} = \{w_1, w_2, \ldots, w_K\}$ in which w_k is the reliability score of the kth source. A higher w_k indicates that the kth source is more reliable, and the values provided by this source are more likely to be accurate.

Given a set of operators $B = \{=, <, >, \neq, \leq, \geq\}$, the DCs are first-order formulas over the repaired result. Each DC takes the form $\varphi : \neg(C_1 \wedge \cdots \wedge C_Z)$, where each clause C_z is of the form $v_{im}^* \odot r(v_{im}^*)$, $r(v_{im}^*)$ represents c_{im}^* or v_{jn}^*, c_{im}^* is a constant, $\odot \in B$, m, n refer to the mth, nth attribute, and $i, j = 1, \ldots, L$[1]. The repaired table $\mathcal{X}^*$ satisfies φ, denoted as $\mathcal{X}^* \models \varphi$, if the values in $\mathcal{X}^*$ meet all the requirements defined in φ.

Example 2. Recall the constraints in Example 1: (1) two persons with the same zip code live in the same city; (2) it is impossible for one person to have his tax greater than his salary. They can be expressed as follows.

$$\varphi_1 : \neg(v_{i1}^* = v_{j1}^* \wedge v_{i2}^* \neq v_{j2}^*), \quad i, j = 1, \ldots, L, \tag{1}$$

$$\varphi_2 : \neg(v_{i3}^* < v_{i4}^*), \quad i = 1, \ldots, L. \tag{2}$$

With the help of DCs, various relationships within one entity or multiple entities are formulated into a uniform format, which can be conveniently incorporated as constraints into the process of conflict resolution. Benefiting from this, the errors can be repaired based on multi-source information for each entity, as well as the trustworthy information from its related entities. Thus, with the consideration of the DCs involvement and source reliability estimation, the data repairing problem is defined as follows.

Problem definition. Given the source information tables $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k\}$ towards a set of entities and a set Σ of DCs defined on the attribute values of these entities, the problem is to find the repaired table $\mathcal{X}^*$ and the source weights $\mathcal{W}$ such that $\mathcal{X}^*$ satisfies Σ as well as gets close to the ground truth.

Furthermore, we formalize this data repairing problem as follows.

$$\begin{aligned} &\min_{\mathcal{X}^*, \mathcal{W}} (\mathcal{X}^*, \mathcal{W}) = \sum_{k=1}^{K} w_k \sum_{l=1}^{L} \sum_{p=1}^{P} d\left(v_{lp}^*, v_{lp}^k\right) \\ &\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) = 1 \end{aligned} \tag{3}$$

[1] Note that in this paper, we are only interested in DCs with at most two entities. DCs involving more entities are less likely in real life and incur a bigger predicate space to find the correct values [7].

$$\mathcal{X}^* \models \varphi, \varphi \in \Sigma,$$

where $d(\cdot)$ is the loss function to measure the distance between the information tables and the repaired result. For continuous data,

$$d\left(v_{lp}^*, v_{lp}^k\right) = \left(v_{lp}^* - v_{lp}^k\right)^2. \tag{4}$$

For categorical data,

$$d\left(v_{lp}^*, v_{lp}^k\right) = \begin{cases} 1 & \text{if} \quad v_{lp}^* \neq v_{lp}^k; \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The basic idea behind the proposed model is that the repaired result needs to be provided by reliable sources as well as satisfying the DCs. To achieve this goal, the repaired values (i.e., v_{lp}^*) should be close to the values (i.e., v_{lp}^k) supported by reliable sources (i.e., w_k is high) and meet all the requirements of φ for each $\varphi \in \Sigma$. Thus, we need to minimize the weighted deviation from the values provided by the conflicting sources to the repaired values, where each source is weighted by its source weight, and the repaired values are constrained by the DCs in Σ.

We prove the hardness of the multi-source data repairing problem defined in Eq. (3) as follows.

Theorem 1. *The multi-source data repairing problem defined in Eq. (3) is NP-complete, even for a small number of DCs.*

Proof. For the DCs $\varphi \in \Sigma$ whose clauses involved with the operator $\odot \in \{=, \neq\}$, we prove the theorem by reducing from the Graph-3-color-ability problem, which is NP-complete [19]. For a graph with n vertices, consider $2n+2$ Skolem constants: $a, b, a_1, \ldots, a_n, b_1, \ldots, b_n$. We use the pair (a_i, b_i) to encode the color of the vertex i: (a, a) stands for 1, (a, b) for 2, and (b, a) for 3. We then have the following DCs: For each vertex i, $a_i = a \vee a_i = b$, $b_i = a \vee b_i = b$, and $a_i = a \vee b_i = a$. For each edge (i, j), $a_i \neq a_j \vee b_i \neq b_j$. Moreover, there is a clause $a \neq b$. These DCs are satisfiable iff. the graph admits a 3-coloring.

For the DCs $\varphi \in \Sigma$ whose clauses are involved with the operator $\odot \in \{<, \leq, >, \geq\}$, we prove the theorem by reducing from the Betweenness problem, which is NP-complete [28]. Given a finite set A of n elements and a collection S of ordered triples (a, b, c) of distinct elements from A, the problem is to determine whether there is a 1-1 function $f : A \to \{1, \ldots, n\}$ such that for each $(a, b, c) \in S$, we have either $f(a) < f(b) < f(c)$ or $f(c) < f(b) < f(a)$. The set A is accordingly represented by the set of indices $\{1, \ldots, n\}$ and the collection S. The Skolem constants are $a_1, \ldots, a_n$. We have the following DCs: For every $i \neq j$, $a_i < a_j \vee a_j < a_i$, encoding that f is 1-1. For every $(i, j, l) \in S$, $(a_i < a_j \wedge a_j < a_l) \vee (a_l < a_j \wedge a_j < a_i)$. Note that the last formula can be rewritten as four DCs defined on every (i, j), (j, i), (j, l) and (l, j). This reduction encodes a 1-1 function from A onto $\{x_1, \ldots, x_n\}$, an n-element subset of the domain. Because the domain is linearly ordered, a 1-1 function f from A to $\{1, \ldots, n\}$ can be defined as $f(i) =$ index of x_i in $\{x_1, \ldots, x_n\}$. These DCs are then satisfiable iff. the function f exists. □

4. Data repairing for inconsistencies and conflicts

In this section, we propose CTD, an iterative approach which considers Conflicts Together with Dependencies, to solve the data repairing problem in Section 3. For the convenience of solving the problem by optimization methods, we first convert the DCs to arithmetic constraints in Section 4.1. We then give solutions for the constrained problem in Section 4.2. Finally, the algorithm CTD is proposed with scalable strategies to ensure its efficiency under massive constraints in Section 4.3.

4.1. Denial constraint transformation

The DCs defined by first-order formulas make it difficult to solve the optimization framework in Eq. (3) with techniques in the optimization methods [5]. Thus, in this section, we attempt to convert first-order formulas to arithmetic constraints for the ease of optimization.

For each DC φ, we first convert its first-order formula $\varphi : \neg(C_1 \wedge \cdots \wedge C_Z)$ into the disjunctive normal form for the convenience of handling each clause separately, i.e. $\varphi : \neg C_1 \vee \cdots \vee \neg C_Z$. As each clause C_z is of the form $v_{im}^* \odot r(v_{im}^*)$, we get:

$$\varphi : v_{im_1}^* \bar{\odot} r\left(v_{im_1}^*\right) \vee \cdots \vee \neg v_{im_Z}^* \bar{\odot} r\left(v_{im_Z}^*\right), \tag{6}$$

for $i = 1, \ldots, L$, where $\bar{\odot}$ is the inverse of the operator $\odot$, and $v_{im_z}^* \bar{\odot} r(v_{im_z}^*)$ denotes the zth clause. For the clauses with different operators $\bar{\odot}$, we add different functions $f_{\bar{\odot}}$, which are shown in Table 4. Each function is characterized with a sign function $sgn(x)$, where

$$sgn(x) = \begin{cases} 1 & \text{if} \quad x > 0; \\ 0 & \text{if} \quad x = 0; \\ -1 & \text{if} \quad x < 0. \end{cases} \tag{7}$$

By assigning different clauses with different functions, we can conclude that $\neg C_z$ is true if $f_{\bar{\odot}}(\neg C_z) \geq 0$. Otherwise, $f_{\bar{\odot}}(\neg C_z) = -1$. With these functions, a set of arithmetic constraints are created towards φ based on Eq. (6):

$$\sum_{z=1}^{Z} f_{\bar{\odot}}\left(v_{im_z}^*, r(v_{im_z}^*)\right) \geq -Z + 1, \tag{8}$$

Table 4
Operator transformation.

Operator($\odot$)	Inverse ($\bar{\odot}$)	Function f	True	False
$=$	$\neq$	$sgn(v^*_{im} - r(v^*_{im}))^2 - 1$	0	-1
$\neq$	$=$	$-sgn(v^*_{im} - r(v^*_{im}))^2$	0	-1
$>$	$\leq$	$-sgn(v^*_{im} - r(v^*_{im}))$	0, 1	-1
$<$	$\geq$	$sgn(v^*_{im} - r(v^*_{im}))$	0, 1	-1
$\geq$	$<$	$sgn(v^*_{im} - r(v^*_{im}))^2 - sgn(v^*_{im} - r(v^*_{im})) - 1$	1	-1
$\leq$	$>$	$sgn(v^*_{im} - r(v^*_{im}))^2 + sgn(v^*_{im} - r(v^*_{im})) - 1$	1	-1

Table 5
Denial constraint classification.

DC	Without constants	With constants
Single entity	Check constraint	Domain constraint
Pairwise entities	FD	CFD

for $i = 1, \ldots, L$, which indicates that for each φ, it is satisfied if at least one clause is true.

We then show the arithmetic constraints generated for some typical DCs. We classify each DC according to two dimensions: (1) the DC defined on the single entity or pairwise entities, (2) the DC contains constants or only has variables. Table 5 shows a representative for each category. Next, we introduce the arithmetic constraints generated for these representatives.

Check constraint. A check constraint φ_c specifies a requirement that must be met between a pair of attributes within an entity. It refers to a DC defined on the single entity and has only variables, each of whose clause is of the form $v^*_{im} \odot v^*_{in}$. According to Eq. (8), a set of arithmetic constraints are created towards φ_c:

$$sgn(v^*_{im} - v^*_{in}) \geq 0, \tag{9}$$

for $i = 1, \ldots, L$, where one clause specific on the mth attribute and the nth attribute exists in φ_c, and $<$ is the operator $\odot$ of the clause.

Domain constraint. A domain constraint φ_d specifies a requirement that must be met between the value of an attribute and a constant within an entity. It refers to a DC defined on a single entity and has constants, each of whose clause is of the form $v^*_{im} \odot c^*_{im}$. According to Eq. (8), a set of arithmetic constraints are created towards φ_d:

$$sgn(v^*_{im} - c^*_{im}) \geq 0, \tag{10}$$

for $i = 1, \ldots, L$, where one clause specific on the mth attribute exists in φ_d, and $<$ is the operator $\odot$ of the clause.

Functional dependency. An FD φ_f defines the relationship among attributes across different entities. It refers to a DC defined on pairwise entities and has only variables. Each clause in such a DC is of the form $v^*_{im} \odot v^*_{jm}$, where $\odot \in \{=, \neq\}$. According to Eq. (8), a set of arithmetic constraints is created towards φ_f:

$$sgn(v^*_{im} - v^*_{jm})^2 - sgn(v^*_{in} - v^*_{jn})^2 \geq 0, \tag{11}$$

for $i, j = 1, \ldots, L, i \neq j$, where two clauses separately associating with the mth, the nth attribute, and $=$, $\neq$ are the operators of the clauses, respectively.

Conditional functional dependency. A CFD φ_{cf} is an extension of an FD φ_f, which has both constants and variables. Each clause of a CFD is of the form $v^*_{im} \odot c^*_{im}$ or $v^*_{im} \odot v^*_{jm}$, where $\odot \in \{=, \neq\}$. According to Eq. (8), a set of arithmetic constraints are created towards φ_{cf}:

$$sgn(v^*_{im} - c^*_{im})^2 + sgn(v^*_{in} - v^*_{jn})^2 - sgn(v^*_{io} - v^*_{jo})^2 \geq 0, \tag{12}$$

for $i, j = 1, \ldots, L, i \neq j$, where three clauses separately associating with the mth, nth, oth attribute, among which the first clause is involved with constant c^*_{im}, and $=$, $=$, $\neq$ are the operators of the clauses, respectively.

Example 3. Considering Example 2, φ_1 is an FD and φ_2 is a check constraint. Two sets of constraints can then be created by Eq. (11) and Eq. (9), respectively.

$$sgn(v^*_{i1} - v^*_{j1})^2 - sgn(v^*_{i2} - v^*_{j2})^2 \geq 0, i, j = 1, \ldots, L, i \neq j, \tag{13}$$

$$sgn(v^*_{i3} - v^*_{i4}) \geq 0, i = 1, \ldots, L. \tag{14}$$

4.2. Proposed solution

Given a set Σ of DCs, the arithmetic constraints are created towards Σ based on Eq. (8). The optimization problem Eq. (3) is rewritten as follows.

$$\min_{\mathcal{X}^*, \mathcal{W}} g(\mathcal{X}^*, \mathcal{W}) = \sum_{k=1}^{K} w_k \sum_{l=1}^{L} \sum_{p=1}^{P} d(v^*_{lp}, v^k_{lp})$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) = 1 \tag{15}$$
$$\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})) \geq -Z_\varphi + 1, i = 1, \ldots, L, \varphi \in \Sigma.$$

For this problem, two sets of unknown variables $\mathcal{X}^*$, $\mathcal{W}$ should be learned together by minimizing the objective function in Eq. (15). However, due to the hardness of the problem, the computational complexity is too high to be afforded. Thus, we choose to iteratively update the values of one set to minimize the objective function while maintaining the values of another set until convergence, which refers to the block coordinate descent approach [38]. To minimize the objective function in Eq. (15), the algorithm iteratively conducts two steps.

Step 1: Source Weights Update. With an initial estimation of the repaired table $\mathcal{X}^*$, the source weights $\mathcal{W}$ are updated by minimizing the objective function with constraints related to $\mathcal{W}$ as follows.

$$\mathcal{W} \leftarrow \arg\min_{\mathcal{W}} g(\mathcal{X}^*, \mathcal{W}) \tag{16}$$
$$\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) = 1,$$

We derive the following equation through Lagrange multipliers for the source weights.

$$w_k = \log\left(\frac{\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{p=1}^{P} d(v^*_{lp}, v^k_{lp})}{\sum_{l=1}^{L}\sum_{p=1}^{P} d(v^*_{lp}, v^k_{lp})}\right). \tag{17}$$

Step 2: Repaired Result Update. In this step, the source weights $\mathcal{W}$ are fixed. The repaired table $\mathcal{X}^*$ is updated by minimizing the objective function with the constraints related to $\mathcal{X}^*$:

$$\mathcal{X}^* \leftarrow \arg\min_{\mathcal{X}^*} g(\mathcal{X}^*, \mathcal{W}) \tag{18}$$
$$\text{s.t.} \quad \sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})) \geq -Z_\varphi + 1, \quad i = 1, \ldots, L, \varphi \in \Sigma,$$

which is the key optimization problem to be solved in the remaining part of this subsection.

Due to the existence of inequality constraints in Eq. (18), we minimize the objective function with the corresponding Karush-Kuhn-Tucker (KKT) conditions [5]. As the KKT conditions differ under the DCs defined on the single entity and pairwise entities, we propose solutions towards these types of DCs.

DCs defined on single entity. When DCs are defined within one entity, i.e., $r(v^*_{im})$ is in form v^*_{in} or c^*_{im}, and for each DC, L constraints are created by varying i from 1 to L. The main conditions are then as follows.

$$\nabla_{v^*_{lp}} L(\mathcal{X}^*, \lambda) = 0 \tag{19}$$

$$-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})) - Z_\varphi + 1 \leq 0 \tag{20}$$

$$\lambda_i^\varphi \left(-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}\left(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})\right) - Z_\varphi + 1\right) = 0 \tag{21}$$

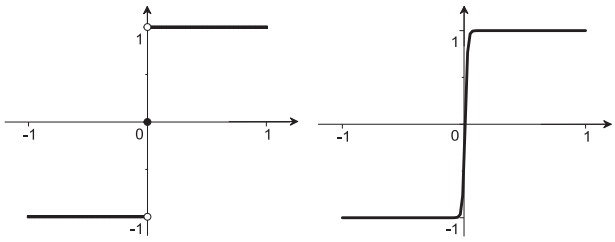
$$\lambda_i^\varphi \geq 0, i = 1, \ldots, L, \varphi \in \Sigma, \tag{22}$$

where

$$L(\mathcal{X}^*, \lambda) = \sum_{k=1}^{K} w_k \sum_{i=1}^{L}\sum_{m=1}^{M} d\left(v^*_{lp}, v^k_{lp}\right) + \sum_{i=1}^{L} \lambda_i^\varphi \left(-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}\left(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})\right) - Z_\varphi + 1\right), \tag{23}$$

$\lambda = \{\lambda_i^\varphi\}_{i=1,L}^{\varphi\in\Sigma}$ are the Lagrangian multipliers.

According to Eq. (21), $\lambda_i^\varphi = 0$ and $-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im_z^\varphi}, r(v^*_{im_z^\varphi})) - Z_\varphi + 1 = 0$ state the following two cases.



(a) $u_1(x)$ (b) $u_2(x)$ with $\theta = 100$

Fig. 1. Curves of $u_1(x)$ and $u_2(x)$.

(1) $\lambda_i^{\varphi} = 0$. In this case, Eq. (20) should be satisfied, indicating that the DCs are inactive. The repaired values $v_{lp}^* \in \mathcal{X}^*$ are then derived by solving Eq. (19), where

$$L(\mathcal{X}^*, \lambda) = \sum_{k=1}^{K} w_k \sum_{i=1}^{L} \sum_{m=1}^{M} d\left(v_{lp}^*, v_{lp}^k\right) \tag{24}$$

For continuous data,

$$v_{lp}^* = \frac{\sum_{k=1}^{K} w_k v_{lp}^k}{\sum_{k=1}^{K} w_k}. \tag{25}$$

For categorical data,

$$v_{lp}^* = \arg\max_{v} \sum_{k=1}^{K} w_k h\left(v, v_{lp}^k\right), \tag{26}$$

where $h(x, y) = 1$ if $x = y$, and 0 otherwise.

(2) $-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v_{im_z^\varphi}^*, r(v_{im_z^\varphi}^*)) - Z_\varphi + 1 = 0$, indicating that the DCs are active. In this case, the values obtained by Eqs. (25) and (26) do not satisfy Eq. (20). Otherwise, in order to satisfy Eq. (21), λ_i^{φ} should be 0, referring to case (1). To simplify the discussion, we denote the set of these values as $\mathcal{X}_a^*$. These values $v_{lp}^* \in \mathcal{X}_a^*$ are then derived by:

$$\begin{aligned} &\mathcal{X}_a^* \leftarrow \arg\min_{\mathcal{X}_a^*} g(\mathcal{X}_a^*, \mathcal{W}) \\ &\text{s.t.} \quad \sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}\left(v_{im_z^\varphi}^*, r(v_{im_z^\varphi}^*)\right) - Z_\varphi + 1 = 0, \end{aligned} \tag{27}$$

where Lagrange multipliers can be adopted to solve the problem.

Remark. To make the constraints continuous and differentiable, which allows us to minimize the objective function under the constraints, we approximate them by replacing the function $u_1(x) = sgn(x)$ with $u_2(x) = \frac{2}{1+\exp(-\theta x)} - 1$. The idea behind the approximation is that we can approximate the function $u_1(x) = sgn(x)$ by function $u_2(x) = \frac{2}{1+\exp(-\theta x)} - 1$ when $x \in (-1, 1)$. Here θ represents the steepness of the curve. We can see from Fig. 1 that $u_2(x)$ is a good approximation for $u_1(x)$.

We then derive the equations to update the repaired result towards typical DCs defined on the single entity, i.e., check constraints and domain constraints, for instance.

Check constraints. When the constraints are created by Eq. (9), the repaired values are updated as follows.

$$v_{lp}^* = \begin{cases} \dfrac{\sum_{k=1}^{K} w_k \left(v_{lm}^k + v_{lp}^k\right)}{2\sum_{k=1}^{K} w_k} & p = n \quad \text{and} \quad \widehat{v}_{lp}^* > \widehat{v}_{lm}^*; \\ \dfrac{\sum_{k=1}^{K} w_k \left(v_{lp}^k + v_{ln}^k\right)}{2\sum_{k=1}^{K} w_k} & p = m \quad \text{and} \quad \widehat{v}_{ln}^* > \widehat{v}_{lp}^*; \\ \widehat{v}_{lp}^* & \text{others}, \end{cases} \tag{28}$$

where $\widehat{v}_{lp}^*$, $\widehat{v}_{lm}^*$, $\widehat{v}_{ln}^*$ are computed according to Eq. (25).

Domain constraints. When the constraints are created by Eq. (10), the repaired values are updated by:

$$v_{lp}^* = \begin{cases} c_{lp}^* & p = m \quad \text{and} \quad \widehat{v}_{lp}^* < c_{lp}^*; \\ \widehat{v}_{lp}^* & \text{others}, \end{cases} \tag{29}$$

where $\widehat{v}_{lp}^*$ are calculated according to Eq. (25).

DCs on pairwise entities. When the DCs are defined among multiple entities, i.e., $r(v^*_{im})$ is in the form v^*_{jn} or c^*_{im}, the L^2 constraints are created at most for each DC by varying i, j from 1 to L. The main conditions are then as follows.

$$\nabla_{v^*_{lp}} L(\mathcal{X}^*, \lambda) = 0 \tag{30}$$

$$-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im^\varphi_z}, r(v^*_{im^\varphi_z})) - Z_\varphi + 1 \le 0 \tag{31}$$

$$\lambda^\varphi_{ij}(-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im^\varphi_z}, r(v^*_{im^\varphi_z})) - Z_\varphi + 1) = 0 \tag{32}$$

$$\lambda^\varphi_{ij} \ge 0, i, j = 1, \ldots, L, \varphi \in \Sigma, \tag{33}$$

where

$$L(\mathcal{X}^*, \lambda) = \sum_{k=1}^{K} w_k \sum_{i=1}^{L}\sum_{m=1}^{M} d(x^*_{im}, x^k_{im}) + \sum_{i=1}^{L} \lambda^\varphi_{ij}\left(-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im^\varphi_z}, r(v^*_{im^\varphi_z})) - Z_\varphi + 1\right), \tag{34}$$

$\lambda = \{\lambda^\varphi_{ij}\}^{\varphi\in\Sigma}_{i,j=1,L}$ are the Lagrangian multipliers.

Similar to the problem defined in the single-entity DCs, updating the repaired result involves two cases.

(1) $\lambda^\varphi_{ij} = 0$, indicating that the DCs are inactive. This makes $L(\mathcal{X}, \lambda)$ the same as case (1) in Section 4.2. Thus, the update of repaired result stays the same with Eqs. (25) and (26).

(2) $-\sum_{z=1}^{Z_\varphi} f_{\bar{\odot}}(v^*_{im^\varphi_z}, r(v^*_{im^\varphi_z})) - Z_\varphi + 1 = 0$, indicating that the DCs are active. Similar to case (2) in Section 4.2, the values obtained by Eqs. (25) and (26) do not satisfy Eq. (31). Thus, we also derive the repaired table $\mathcal{X}^*_a$ by solving Eq. (27). However, the difference from the single-entity case is that the constraints towards pair-wise entities make the relationship among the attribute values more complex. More specifically, the value of the attribute of one entity may influence the value of the corresponding attribute of another entity. Thus, the calculation of the optimal correct values will be too costly to be afforded [12].

To deal with this issue, we only update the repaired values involved in the functions $f_{=}(v^*_{im^\varphi_z}, r(v^*_{im^\varphi_z}))$ with the operator "$\neq$" while maintaining the update of the others' values according to case (1), which corresponds to the modifications performed only on the right-hand of the constraints [4] (e.g., FDs) in the area of data repairing.

For instance, we derive the repaired value update towards the typical DCs defined on the pairwise entities, i.e., the CFDs. Note that the FDs is a special case of the CFDs, and thus the equation to update the repaired values in the CFDs can also be used for the FDs.

Conditional functional dependency. When the constraints are created by Eq. (12), the repaired values are updated according to:

$$v^*_{lp} = \begin{cases} \arg\max_v \sum_{k=1}^{K} w_k (h(v, v^k_{io}) + h(v, v^k_{jo})) & \text{case (a);} \\ \widehat{v}^*_{lp} & \text{case (b).} \end{cases} \tag{35}$$

Case (a) refers to $l = i$ or $l = j$, $p = o$, where $\widehat{v}^*_{im} = c^*_{im}$, $\widehat{v}^*_{in} = \widehat{v}^*_{jn}$, $v^*_{io} \neq v^*_{jo}$, and case (b) refers to the others. Here $\widehat{v}^*_{im}, \widehat{v}^*_{jm}, \widehat{v}^*_{jn}, \widehat{v}^*_{io}, \widehat{v}^*_{jo}, \widehat{v}^*_{lp}$ are calculated by Eq. (26).

4.3. The CTD algorithm

With the solution proposed in Section 4.2, we summarize the pseudo code of our data repairing algorithm CTD in Algorithm 1. We start with an initial estimation of the source weights (Line 1) and then iteratively conduct the source weights update and repaired result update steps until convergence (Lines 2–12). The convergence criterion is that the decrease in the objective function is small enough compared with the previous iterations, which has the same setting as [22,23].

Example 4. Consider Example 1. Using the same databases and constraints, the CTD updates the source weights and the repaired result until the convergence criterion is satisfied. We show the repaired results of the CTD for the first three iterations in Table 6.

Take the inference of Mike's salary and Mike's tax as an example. Suppose that the source weights are $\mathcal{W} = \{0.344, 2.83 * 10^{-2}, 10^{-5}\}$ in the third iteration. In case (1), according to Eq (25), the repaired values for Mike's salary and Mike's tax are

Algorithm 1: CTD Algorithm.

Input: Data from K sources, a set Σ of DCs
Output: The repaired table $\mathcal{X}^* = \{v^*_{lp}\}^{L,M}_{l=1,p=1}$, source weights $\mathcal{W} = \{w_1, \ldots, w_K\}$

1: Initialize the source weights $\mathcal{W}$
2: **while** convergence criterion is not satisfied **do**
3: **for** $l \leftarrow 1$ to L **do**
4: **for** $p \leftarrow 1$ to M **do**
5: Calculate $\widehat{v}^*_{lp}$ according to Eqs. (25) and (26)
6: **if** $\widehat{v}^*_{lp}$ does not satisfy Eqs. (20) and (31) **then**
7: Update v^*_{lp} according to Eq. (27)
8: **else**
9: Update $v^*_{lp} = \widehat{v}^*_{lp}$
10: **for** $k \leftarrow 1$ to K **do**
11: Update the weight of the kth source w_k by Eq. (17)
12: **return** $\mathcal{X}^*$ and $\mathcal{W}$

Table 6
The repaired results for the first three iterations.

	Iteration 1				Iteration 2				Iteration 3			
Entity	zip	city	salary	tax	zip	city	salary	tax	zip	city	salary	tax
Bob	10002	NYC	60000	4983	10002	NYC	60000	5000	10002	NYC	60000	5000
Kate	10002	NYC	32500	3250	10002	NYC	35000	3500	10002	NYC	35000	3500
Mike	14221	BUF	24333	11933	14221	BUF	24051	24051	14221	BUF	24051	24051

"20227" and "27876", respectively. It can be easily inferred that $20227 < 27876$, which violates constraint (2). The repaired values are then updated to "24051" and "24051" according to case (2), i.e., Eq. (28) for the check constraints. Compared to the ground truth in Table 2, the repaired result gets more and more accurate with more iterations. The CTD converges after three iterations and performs better than the CRH and HoloClean.

To show the properties of the algorithm, we first prove its convergence and analyze the time complexity. we then propose two scalable strategies to ensure its efficiency when faced with massive constraints.

Finally, we discuss two important issues to make the algorithm practical, i.e., missing values and noisy DCs.

Convergence. Using several types of loss functions and DCs discussed in this paper, we prove the convergence of the CTD as follows.

Theorem 2. *When Eq. (9) or/and Eq. (10) is/are adopted as the constraints and Eq. (4) is used as the loss function, the convergence of the CTD algorithm is guaranteed.*

Proof. For the optimization problem in Eq. (15), it can be easily inferred that the unique minimum with respect to $\mathcal{W}$ is achieved when $\mathcal{X}^*$ is fixed and a unique minimum with respect to $\mathcal{X}^*$ is achieved when $\mathcal{W}$ is fixed [5]. Therefore, according to the proposition on the convergence of the block coordinate descent [38], CTD will converge to a stationary point of the proposed problem. □

Time Complexity. In each iteration, CTD first computes the repaired value for each attribute of each entity in case (1), whose time complexity is $\mathcal{O}(KLM)$, where K is the number of sources, L is the number of entities, and M is the number of attributes (Lines 1–5). For case (2), it needs at most $\mathcal{O}(L^2M)$ if it is constrained by a pairwise level DC (Lines 6–9). The time complexity of the source weights update is also $\mathcal{O}(KLM)$, which needs to traversal the whole data from the K sources (Lines 10–12). Thus, the running time is at most $\mathcal{O}(rKLM + rL^2M)$, where r is the number of iterations. When L is large, CTD will become too costly to scale well for a large amount of data. To handle this issue, we discuss the scalable strategies in the following part to make the CTD more practical.

Scalable Strategies The running time of the CTD will become relatively large when the DCs are defined on pairwise entities (in Algorithm 1 Line 7), as they constrain the relationship between every pairwise entity in the truth table. Thus, we discuss two strategies to reduce the cost for such DCs.

Optimal Grouping. Instead of repairing the attribute values for each pairwise entity, we group the entities sharing the same values on a part of the attributes (i.e., the attributes involved in the functions $f_{\neq}(v^*_{im^{\varphi}_z}, r(v^*_{im^{\varphi}_z}))$ with the operator " $=$ "). We then update the values of the other attributes of these entities once.

Domain Pruning of Candidate Values. During the process of searching, when the candidate values for an entity are too many, it could be hard to find the best repair which minimizes the objective function in Eq. (27). Thus, we prune the domain of the candidate values. Suppose for $\mathcal{X}^*_a = \{v^*_{io}, v^*_{jo}\}$, C combinations $\{v_{io_c}, v_{jo_c}\}^C_{c=1}$ of the candidate values

from $\{v_{io}^k\}_{k=1}^K$ and $\{v_{jo}^k\}_{k=1}^K$ satisfy Eqs. (20) and (31). To minimize Eq. (27), the probability that $\{v_{io_c}, v_{jo_c}\}$ are the values is then $\sum_{k=1}^K w_k \sum_{l\in\{i,j\}} h(v_{lo_c}, v_{lo}^k)$. Here we denote such probability for combinations as $p_1, \ldots, p_C$. Clearly, if $p_{\max} = \max\{p_1, \ldots, p_c\} > |\mathcal{X}_a^*| - (p_1 + \cdots + p_c)$, the correct values should be $\{v_{io_{\max}}, v_{jo_{\max}}\}$, and the probabilities for the $(c+1)$th to the Cth combinations no longer need to be calculated.

With such scalable strategies, the time complexity of the repaired result calculation in case (1) as well as the source weight calculation is unchanged, which are both $\mathcal{O}(rKLM)$, where r is the number of iterations, K is the number of sources, L is the number of entities, and M is the number of attributes. For case (2), updating the repaired values under a pairwise DC only needs $\mathcal{O}(rLM)$. This is because we group the tuples according to the repaired values of a part of attributes and update the repaired values of the other attribute once, which costs $\mathcal{O}(LM)$ using the hash table. In summary, the time complexity of CTD is $\mathcal{O}(rKLM)$, which is much less than $\mathcal{O}(rKLM + rL^2M)$, as in real life, $L \gg K$.

Missing Values. For the sake of simplicity, in the proposed framework of Eq. (3), we assume that the information of all the attributes of all the entities is provided by all the sources. It can be easily modified to handle missing values when different sources provide information for different subsets of the entities on different subsets of the attributes. When the number of values provided by different sources is quite different, we can normalize the overall distance of each source by the number of its providing values. Note that the missing values in the source information table $\{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ do not affect the operation $(=, \neq, \leq, \geq, >, <)$ between the repaired values in the repaired table $\mathcal{X}^*$. According to the repaired result calculation approach (Eqs. (25) and (26)) in case (1), the repaired value is calculated based on the corresponding values provided by all the sources. Thus, the repaired value can be obtained as long as one source provides its value. With the repaired values, the operation can then be normally applied.

Noisy DCs. Another important issue is the correctness of the DCs. In the proposed framework Eq. (3), the DCs are used to repair the multi-source noisy data. If the DCs are imprecise, it will be difficult to guarantee the accuracy of the repaired result. However, DCs could also be dirty in real-world scenarios [34]. To solve this issue, for the case when both DCs and data are imprecise, the multi-source data repairing problem (Eq. (3)) can be adjusted to a θ-tolerant multi-source data repairing problem, with tolerance on the DC variances: Given the source information tables $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k\}$ towards a set of entities and a set Σ of DCs defined on the attribute values of these entities, the problem is to find the repaired table $\mathcal{X}^*$ and the source weights $\mathcal{W}$ such that (1) $g(\mathcal{X}^*, \mathcal{W})$ is minimized, and (2) $\mathcal{X}^*$ satisfies Σ', for some constraint variant Σ' of Σ with $\Theta(\Sigma, \Sigma') \leq \theta$. Different sets of Σ' of the DC variants can be generated [34], and the repaired result update approach can be applied to each Σ'. The repaired table $\mathcal{X}'$ under Σ' with the minimum $g(\mathcal{X}', \mathcal{W})$ is then the final result $\mathcal{X}^*$.

5. Experiments

In this section, we evaluate the proposed method using four real-world datasets. The experimental results clearly demonstrated the advantages of the proposed method by considering both the source weight estimation and DCs in data repairing. We first discuss the experimental setup in Section 5.1. We then show the experimental results in terms of effectiveness and efficiency in Sections 5.2 and 5.3, respectively.

5.1. Experimental setup

Datasets. To comprehensively test the proposed methods, we conducted extensive experiments on four real-world datasets.

Weather. The weather data [23,26], obtained over a two-month period starting from October 2013, consists of the weather data from nine sources and the ground truths. The data have three attributes: High Temperature (HT), Low Temperature (LT) and Weather Condition (WC), among which the first two are continuous, and the last is categorical. A domain constraint is defined as; it is impossible that the WC is Snowy while LT is more than 55°F[2]. *We use this dataset to test how effective the* CTD *is when the denial constraint is a domain constraint.*

Flight. The flight data [24], obtained over a one-month period starting from December 2011, consists of departure and arrival information for 1200 flights from 38 sources. The ground truths are also available. We preprocess the data to convert the time information into minutes and treat it as a continuous type. A check constraint is defined as; it is impossible that the actual departure time is earlier than (i.e., less than) the scheduled departure time. *We use this dataset to test how effective the* CTD *is when the denial constraint is a check constraint.*

Stock. The stock data [24], obtained every work day in July 2011, consists of 1000 stock symbols from 55 sources with the ground truths also provided. We treat the data as the categorical type and a set of CFDs is defined as; two stock records with the same symbol have the same 52wk High, 52wk Low and EPS. The constant table contains the 52wk High, 52wk Low and EPS of 161 stocks in the same period from each official website. *We use this dataset to test how effective the* CTD *is when the denial constraint is a conditional functional dependency.*

Restaurant. The restaurant data [40,42] consists of a list of 6324 restaurants' information from 5 websites (sources) with the ground truths also provided. The data has 5 categorical attributes: Restaurant Name (RN), Building Number (BN), Street Name (SN), Zip Code (ZC), and Phone Number (PN). An FD is defined as; two restaurants with the same SN

[2] The setting number is reasonable as the snow will change over to rain if LT $\geq$ 55°F

Table 7
Performance comparison on weather and flight datasets.

	Weather						Flight	
Methods	Accuracy	Precision	Recall	F1-score	MAE	RMSE	MAE	RMSE
CTD	**0.7594**	**0.7415**	**0.5610**	**0.6387**	**4.8526**	**7.8648**	**23.2841**	**111.4840**
CATD	0.7043	0.6970	0.4914	0.5764	4.9453	8.0207	24.5089	115.7255
CRH	0.7318	0.7379	0.5110	0.6038	4.8529	7.8661	25.9699	116.4947
GTM	N/A	N/A	N/A	N/A	5.3220	8.2676	24.1043	112.1847
HoloClean	0.5414	0.4010	0.2504	0.3083	5.5875	8.5996	0.0[#]	0.0[#]
HoloClean*	0.5464	0.4095	0.2551	0.3144	5.6326	8.6352	0.0[#]	0.0[#]

[#] HoloClean accidentally terminated without results.

and BN have the same ZC. *We use this dataset to test how effective the* CTD *is when the denial constraint is a functional dependency.*

Algorithms. For the proposed methods, we test the CTD with two scalable strategies. The baseline methods are listed as follows.

Truth discovery baselines.

CRH [23]: This approach models the conflict resolution problem by solving an optimization problem. The solution consists of a two-step iterative procedure to update the truths and source weights.

CATD [22]: This approach also models the conflict resolution problem by solving an optimization problem which extends the source weights with confidence intervals to account for the sparsity in the source observations.

GTM [45]: This is a Bayesian probabilistic approach designed for continuous data, while other methods can apply to different types of data. However, as an important method in truth discovery, we still include it in the comparison. *Constraint-based data repairing baseline.*

HoloClean [32]: This is a constraint-based data repairing approach driven by a probabilistic inference, which allows users to flexibly define multiple types of DCs. When considering the candidate repaired value for a target entity's attribute c, the value $v_{c'}$ of the attribute c' for the same entity is taken into account as evidence. The probability that all the values in the domain of c that co-occur with the value $v_{c'}$ is calculated as:

$$Pr[v|v_{c'}] = \frac{\#(v, v_{c'}) \text{ appear together in } D}{\#v_{c'} \text{ appears in } D}.$$

In the experiments, extra DCs are defined to ensure the unique attribute value for each entity. HoloClean is shown to outperform the other data repairing methods. Thus, we chose to compare the CTD against this method.

HoloClean*: This is a HoloClean version with the consideration of source weight. More specifically, when considering the candidate repaired value for a target entity's attribute c, the probability that all the values in the domain of c co-occurring with the value $v_{c'}$ is calculated as:

$$Pr[v|v_{c'}] = \frac{\sum_{S_k \text{ supported } (v,v_{c'})} w_k}{\sum_{S_k \text{ supported } v_{c'}} w_k},$$

in which the source weights $\mathcal{W} = \{w_1, \ldots, w_K\}$ are obtained from CRH [23].

Evaluation measures. In this experiment, we focus on both the categorical and continuous data. To evaluate the performance of the proposed method and the baseline methods, we adopt the following measures for these two data types.

Categorical data. We use **Accuracy, Precision, Recall**, and **F1-score** as the performance measures of an approach. Let *truth* be the set of erroneous attribute values in $\{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ and *found* be the set of repaired attribute values according to $\mathcal{X}^*$. Accuracy is computed as the percentage of the approach's outputs $\mathcal{X}^*$ which are the same as the ground truths. Precision is calculated as $precision = \frac{|truth \cap found|}{|found|}$, denoting the proportion of corrected attribute values to the number of all the attribute values that are repaired. Recall is calculated as $recall = \frac{|truth \cap found|}{|truth|}$, representing the proportion of corrected attribute values to the number of all the erroneous attribute values. The F1-score is the harmonic mean of Precision and Recall.

Continuous data. We calculated the following metrics on the approach's outputs by comparing them with the ground truths, Mean of Absolute Error (**MAE**) and Root of Mean Squared Error (**RMSE**) [26]. MAE uses the L^1-norm distance that penalizes more on smaller errors, while RMSE adopts the L^2-norm distance that gives more penalty to the bigger errors.

5.2. Effectiveness evaluation

To study the effectiveness of the proposed method, we first show the overall performance of the proposed method as well as the baseline methods. We then study the effectiveness from the following perspectives: (1) the effect of the coverage rate of sources, (2) the effect of the number of sources, (3) the effect of the number of DCs, and (4) robustness evaluation.

Performance comparison. Tables 7 and 8 summarize the results for all the methods using the four real-world datasets. In terms of effectiveness, the proposed CTD method achieves the best performance on all datasets, and the improvement

Table 8
Performance comparison on stock and restaurant datasets.

	Stock				Restaurant			
Methods	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
CTD	**0.9064**	**0.7282**	**0.5840**	**0.6482**	**0.9440**	**0.8333**	**0.4724**	**0.6030**
CATD	0.8347	0.5209	0.4277	0.4697	0.9398	0.8013	0.4314	0.5609
CRH	0.8947	0.6924	0.5576	0.6177	0.9398	0.8013	0.4314	0.5609
GTM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HoloClean	0.0[#]	0.0[#]	0.0[#]	0.0[#]	0.9366	0.6646	0.3886	0.4904
HoloClean*	0.0[#]	0.0[#]	0.0[#]	0.0[#]	0.9403	0.8051	0.4492	0.5767

[#] HoloClean accidentally terminated without results.

is significant. For instance, for the *Weather* dataset, compared to the best CRH baseline, CTD's Accuracy, Precision, Recall, and F1-score increased by 2.76%, 0.36%, 5%, and 3.49%, respectively; For the *Flight* dataset, compared with the best baseline GTM, the MAE and the RMSE of the CTD decreased by 3.4% and 0.62%, respectively. Among the baseline methods, HoloClean simply aggregates the multi-source information without considering the source reliability. Thus, it has the worst performance. HoloClean* resolves the inconsistencies by considering the source reliability, and thus the performance is better than HoloClean. However, the improvement is not obvious. Although the source reliability is adopted as the prior knowledge to generate repair candidates, its core probabilistic model treats each entity's attribute as an irrelevant individual and builds the relationship through the given constraints. During this process, the influence of the source weights is not considered. GTM estimates the source reliability only by continuous data which may not have sufficient information, and thus leads to a bad performance for the *Weather* dataset. CATD and CRH are more appropriate for the tasks with various data types. Our proposed CTD method repairs the data based on both the source weight estimations and DCs. By considering both inconsistencies and conflicts existing in multi-source data, it achieves the best performance.

The effect of sources' coverage rate. We first compared the effectiveness of the different approaches under various coverage rates of the sources. The rate is defined as the percentage of the number of sources providing information for each entity. The results are shown in Fig. 2. It can be seen that the proposed CTD method achieves a significantly lower MAE, RMSE and higher Accuracy, Precision, Recall, and F1-score in most cases. Moreover, when the coverage rate is low (e.g., 0.2), CTD performs much better than the other baselines. For instance, for the *Flight* dataset, compared to the best CATD baseline, the MAE and the RMSE of CTD decreased by 2.9% and 2.8%, respectively. The results confirm the idea proposed in this paper, i.e., the less reliable information we have for some entities (i.e., reliable sources provide no information), the greater the DCs can help to find their correct values.

The effect of the number of sources. To explore the influence of the number of sources on the overall performance, we also studied the effectiveness of the proposed method by varying the number of sources. The results are shown in Fig. 3.

With the increase in the number of sources, the overall trend of the RMSE and MAE of all the methods decreases, and the overall trend of the Accuracy, Precision, Recall, and F1-score of all the methods increases. Among these methods, the proposed CTD method performs the best in most cases. However, the improvement is not obvious compared to the other baseline methods, especially under the condition with fewer sources.

The reason is that when the number of sources is limited (i.e., reliable sources may not be available), the effectiveness of discovering the correct values for most entities cannot be ensured, which also influences the performance of the CTD.

The effect of the number of constraints. To show the improvement of the effectiveness regarding the number of DCs, we studied the effectiveness of the proposed CTD method by varying the number of CFDs on the *Stock* dataset. The results are shown in Fig. 4.

It can be seen that with the increase in the number of constraints, the Accuracy, Precision, Recall, and F1-score of CTD increase, indicating the power of applying DCs during data repairing.

Additionally, as we discussed above, on the other three datasets, the proposed method achieves significant improvement by only defining one DC. Thus, it can be inferred that the performance of the proposed method can be further improved when more DCs are available.

Robustness evaluation. To further show the robustness of the proposed algorithm CTD, we randomly split each dataset 10 times using a fix splitting ratio and report the average performance with standard deviations. The results are shown in Fig. 5. It can be seen that with the increase in the splitting ratio, the Accuracy, Precision, Recall, and F1-score of CTD arise, and the MAE and RMSE decrease in most cases. As for the standard deviation, when the splitting ratio is low/high, the standard deviation tends to be relatively large/small. The reason is that CTD can find more correct values when sufficient information is available (i.e., splitting ratio is high), which also leads to a stable performance. When the splitting ratio is low, the performance fluctuates depending on the limited data. Specifically, if the data contain enough correct values, more erroneous values could be repaired through the DCs and reliable sources. Otherwise, the performance tends to decrease. Overall, the standard deviation is acceptable. For categorical data, the standard deviation is less than 0.1 in most cases. For continuous data, the standard deviation from the average performance is also small. The experimental results indicate the robustness of the CTD.

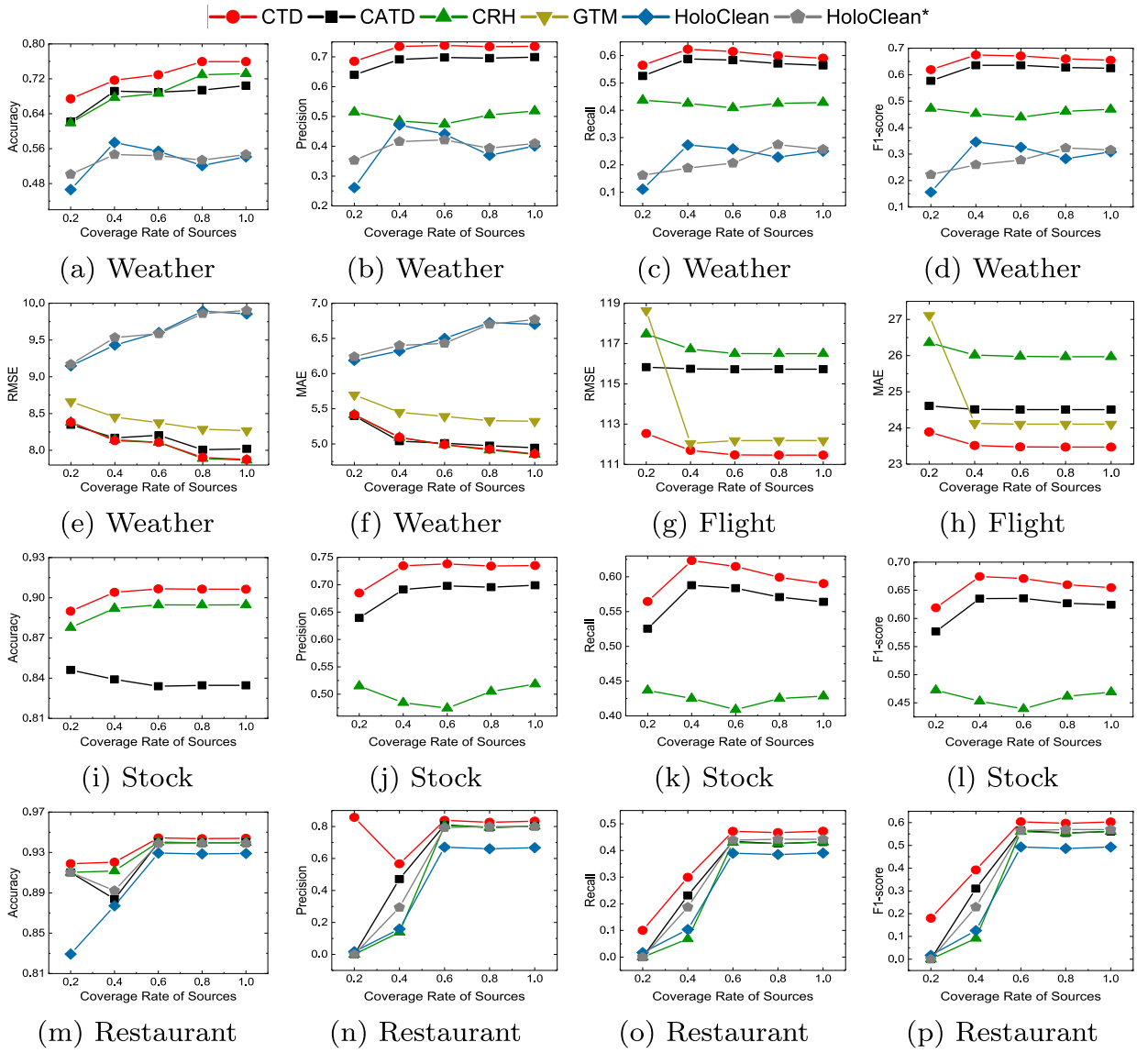


Fig. 2. Performance comparison w.r.t. varying coverage rate.

5.3. Efficiency evaluation

In this part, we evaluate the efficiency of the proposed method and the baseline methods. We first explore their convergence and then show their running time.

Convergence speed. As the proposed CTD method takes advantage of an iterative procedure, we test the convergence of the CTD as well as the iterative baseline methods, CATD and CRH. Fig. 6 shows the change in the objective value of each method with respect to each iteration for the *Stock* and *Restaurant* datasets. We can see that the objective values of the CTD and CRH decrease fast within the first three iterations and then reach a stable stage. The reason for the fast converging speed of the CTD is that the proposed objective function in Eq. (3) is biconvex. Thus, we need to alternatively optimize each variable. Due to the large gradients in the first three iterations, the variables dramatically change, resulting in the fast decrease in the objective value. The experimental result indicates that the CTD converges as quickly as the baseline methods. We omit the results for the *Weather* and *Flight* datasets, as they have similar performances.

Running time. Table 9 summarizes the running time for all the methods of the four real-world datasets. We observe that HoloClean and HoloClean* are inefficient, as they use a DeepDive framework which takes extra time to process the data. More specifically, HoloClean* takes a longer time than HoloClean due to the extra calculation according to the source weights. The proposed method CTD is slightly slower than the best baseline CRH, as CTD needs extra time to process the

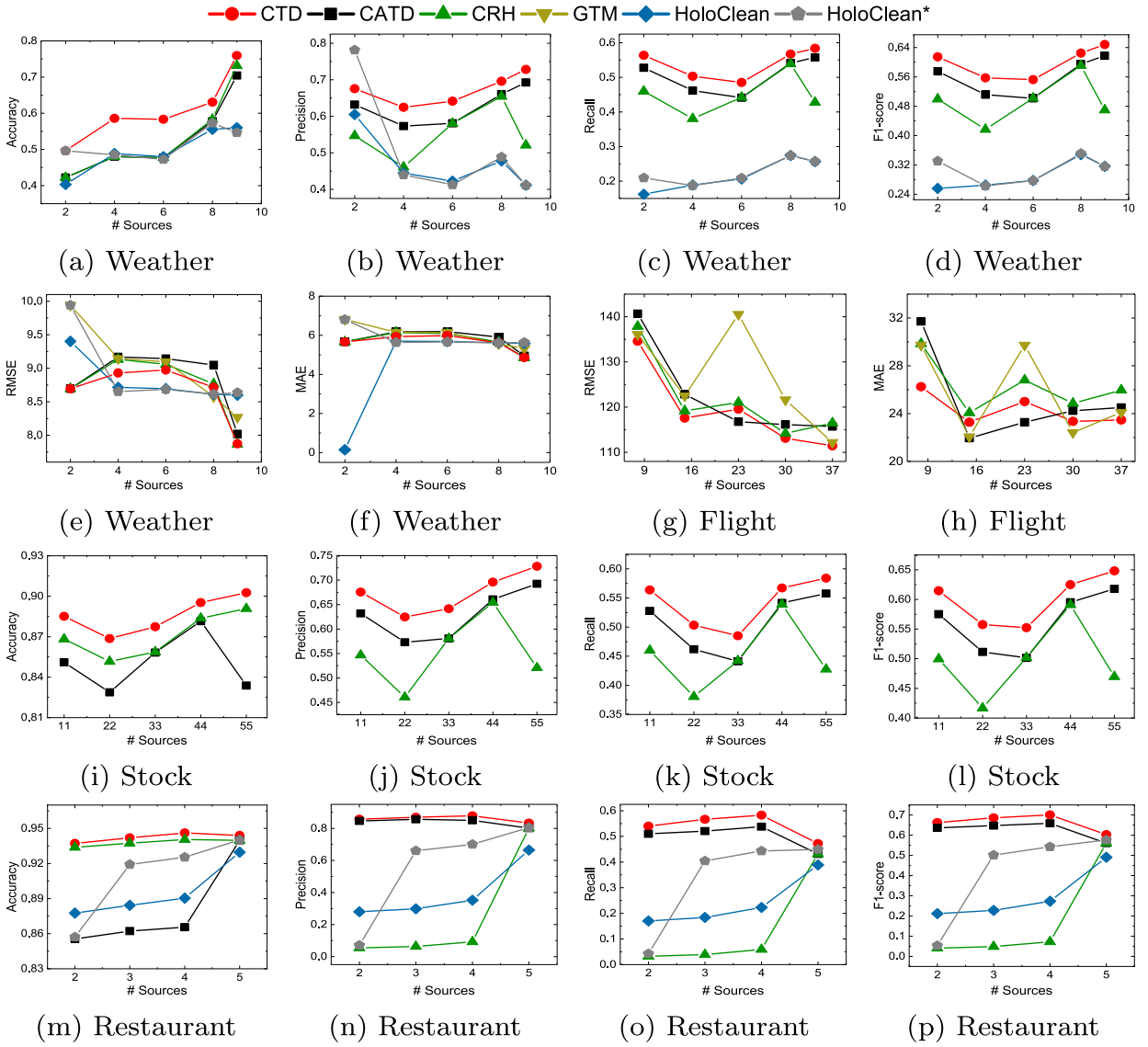


Fig. 3. Performance comparison w.r.t. # sources.

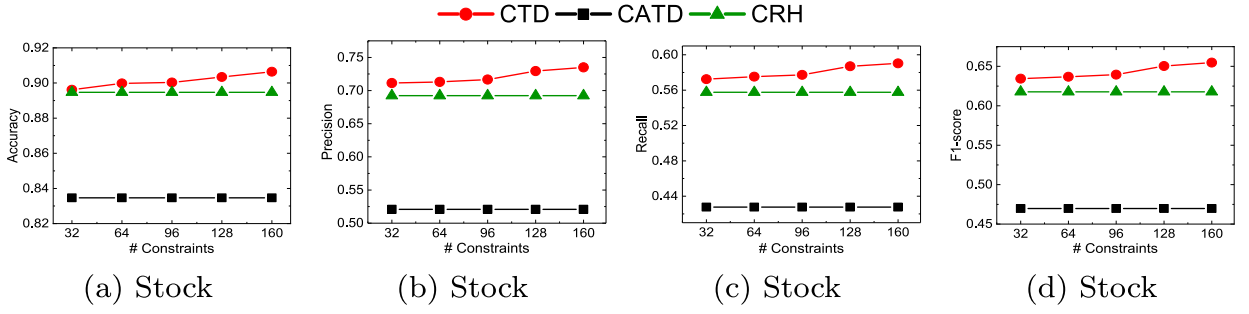


Fig. 4. Performance comparison w.r.t. # constraints.

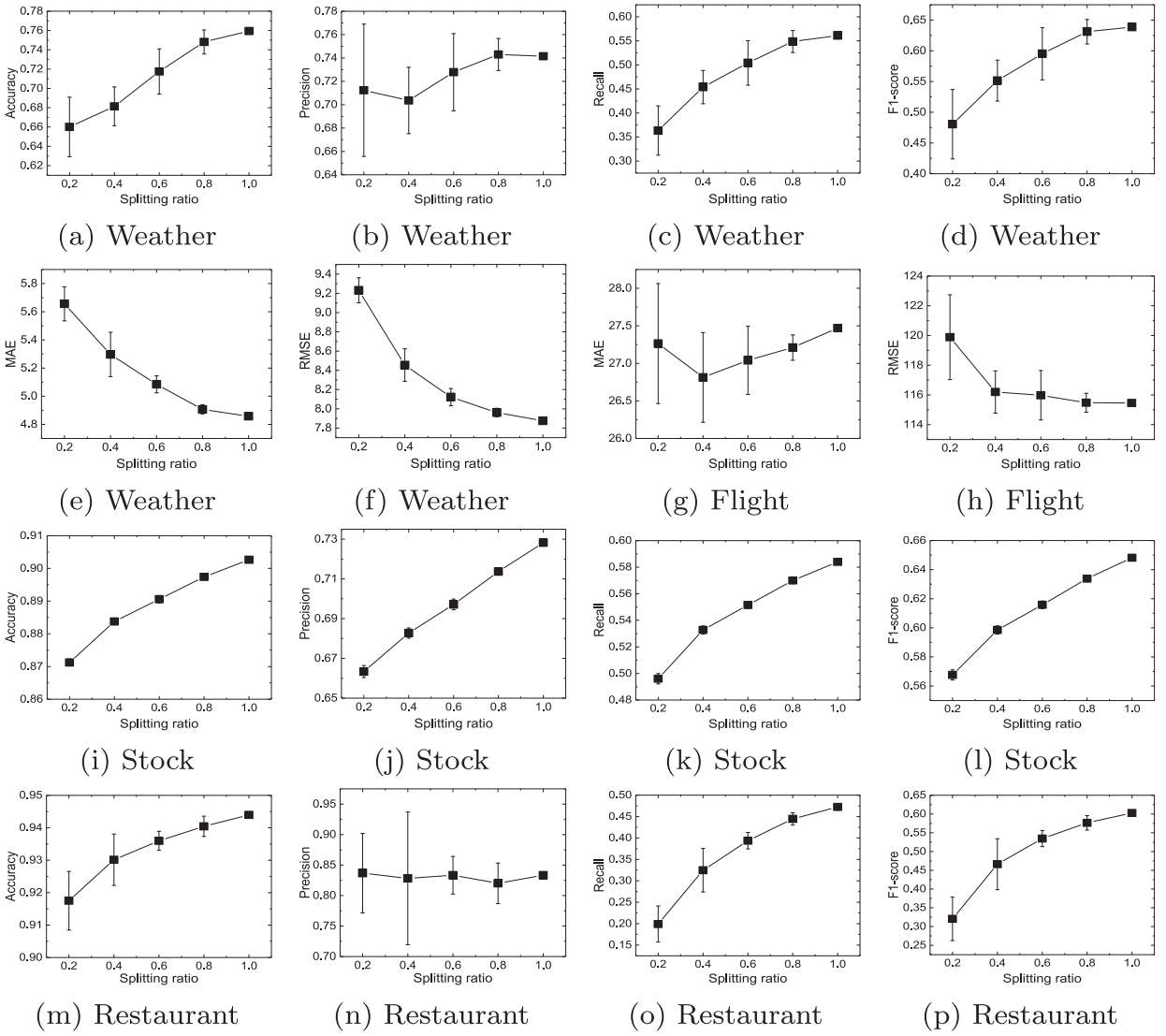


Fig. 5. Robustness evaluation.

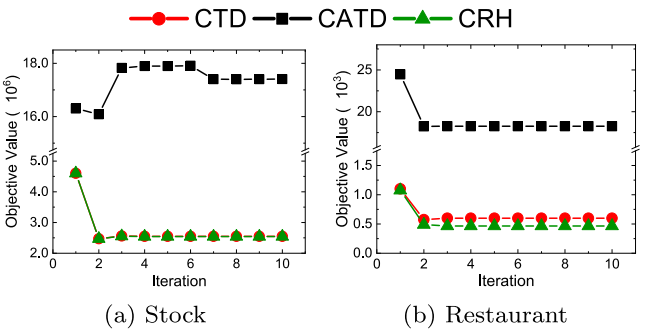


Fig. 6. Convergence speed.

Table 9
Running time (sec).

Methods	Weather	Flight	Stock	Restaurant
CTD	0.3638	64.9816	127.6632	5.8098
CATD	0.6128	135.3192	548.3	17.4143
CRH	0.3533	61.4282	129.8724	5.5681
GTM	0.3673	68.8928	N/A	N/A
HoloClean	28.2849	0.0[#]	0.0[#]	162.4258
HoloClean*	31.4736	0.0[#]	0.0[#]	175.3412

[#] HoloClean accidentally terminated without results.

DCs. Considering its improvement in effectiveness, the CTD does not sacrifice too much on its efficiency, and it is still faster than CATD, GTM, HoloClean, and HoloClean*.

6. Conclusion

In this paper, we propose a novel data repairing approach for both inconsistencies and conflicts. We formulate the data repairing problem as an optimization problem and create constraints for the DCs. To solve this problem, we derive a two-step iterative algorithm called the CTD to iteratively update the source weights and the repaired results, and four concrete cases using different classes of DCs as examples. We also develop two optimal strategies to ensure the scalability of the proposed algorithm. The experimental results of the real-world datasets verify the effectiveness and efficiency of the proposed framework under the different challenging scenarios. In the future, we plan to extend the framework so that it can automatically discover DCs for data repairing.

Declaration of Competing Interest

None.

Acknowledgments

This paper was partially supported by NSFC grant U1509216, U1866602, 61602129, NSF IIS-1553411, NSF-IIS 1747614 and Microsoft Research Asia.

References

[1] L. Bertossi, S. Kolahi, L.V. Lakshmanan, Data cleaning and query answering with matching dependencies and matching functions, Theory Comput. Syst. 52 (3) (2013) 441–482.
[2] T. Bleifuß, S. Kruse, F. Naumann, Efficient denial constraint discovery with hydra, PVLDB 11 (3) (2017) 311–323.
[3] J. Bleiholder, F. Naumann, Data fusion, ACM Comput. Surv. 41 (1) (2008) 1–41.
[4] P. Bohannon, W. Fan, M. Flaster, R. Rastogi, A cost-based model and effective heuristic for repairing constraints by value modification, in: Proc. of SIGMOD, 2005, pp. 143–154.
[5] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[6] J. Chomicki, J. Marcinkowski, Minimal-change integrity maintenance using tuple deletions, Inf. Comput. 197 (1–2) (2005) 90–121.
[7] X. Chu, I.F. Ilyas, P. Papotti, Discovering denial constraints, PVLDB 6 (13) (2013) 1498–1509.
[8] X. Chu, I.F. Ilyas, P. Papotti, Holistic data cleaning: Put violations into context, in: Proc. of ICDE, 2013.
[9] G. Cong, W. Fan, F. Geerts, X. Jia, S. Ma, Improving data quality: Consistency and accuracy, in: Proc. of VLDB, 2007, pp. 315–326.
[10] X.L. Dong, L. Berti-Equille, D. Srivastava, Truth discovery and copying detection in a dynamic world, PVLDB 2 (1) (2009) 562–573.
[11] W. Fan, Dependencies revisited for improving data quality, in: Proc. of PODS, 2008, pp. 159–170.
[12] W. Fan, Data quality: from theory to practice, SIGMOD Record 44 (3) (2015) 7–18.
[13] W. Fan, F. Geerts, Foundations of data quality management, Synth. Lect. Data Manage. 4 (5) (2012) 1–217.
[14] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, TODS 33 (2) (2008) 6.
[15] W. Fan, F. Geerts, N. Tang, W. Yu, Inferring data currency and consistency for conflict resolution, in: Proc. of ICDE, 2013, pp. 470–481.
[16] W. Fan, X. Jia, J. Li, S. Ma, Reasoning about record matching rules, PVLDB 2 (1) (2009) 407–418.
[17] W. Fan, J. Li, S. Ma, N. Tang, W. Yu, Towards certain fixes with editing rules and master data, PVLDB 3 (1–2) (2010) 173–184.
[18] A. Galland, S. Abiteboul, A. Marian, P. Senellart, Corroborating information from disagreeing views, in: Proc. of WSDM, 2010, pp. 131–140.
[19] M.R. Garey, D.S. Johnson, L.J. Stockmeyer, Some simplified np-complete graph problems, Theor. Comput. Sci. 1 (3) (1976) 237–267.
[20] J. Gryz, B. Schiefer, J. Zheng, C. Zuzarte, Discovery and application of check constraints in DB2, in: Proc. of ICDE, 2001, pp. 551–556.
[21] S. Kolahi, L.V. Lakshmanan, On approximating optimum repairs for functional dependency violations, in: Proc. of ICDT, 2009, pp. 53–62.
[22] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, J. Han, A confidence-aware approach for truth discovery on long-tail data, PVLDB 8 (4) (2014) 425–436.
[23] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, J. Han, Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation, in: Proc. of SIGMOD, 2014, pp. 1187–1198.
[24] X. Li, X.L. Dong, K.B. Lyons, W. Meng, D. Srivastava, Truth finding on the deep web: is the problem solved? PVLDB (2013).
[25] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, J. Han, A survey on truth discovery, Proc. of SIGKDD 17 (12) (2015) 1–16.
[26] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, J. Han, On the discovery of evolving truth, in: Proc. of SIGKDD, 2015, pp. 675–684.
[27] C. Mayfield, J. Neville, S. Prabhakar, ERACER: a database approach for statistical inference and data cleaning, in: Proc. of SIGMOD, 2010, pp. 75–86.
[28] J. Opatrny, Total ordering problem, SIAM J. Comput. 8 (1) (1979) 111–114.
[29] J. Pasternack, D. Roth, Knowing what to believe (when you already know something), in: Proc. of ICCL, 2010, pp. 877–885.
[30] J. Pasternack, D. Roth, Making better informed trust decisions with generalized fact-finding, in: Proc. of IJCAI, 2011, pp. 2324–2329.

[31] R. Pochampally, A. Das Sarma, X.L. Dong, A. Meliou, D. Srivastava, Fusing data with correlations, in: Proc. of SIGMOD, 2014, pp. 433–444.
[32] T. Rekatsinas, X. Chu, I.F. Ilyas, C. Ré, HoloClean: holistic data repairs with probabilistic inference, PVLDB 10 (11) (2017) 1190–1201.
[33] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, C. Ré, SLiMFast: guaranteed results for data fusion and source reliability, in: Proc. of SIGMOD, 2017, pp. 1399–1414.
[34] S. Song, H. Zhu, J. Wang, Constraint-variance tolerant data repairing, in: Proc. of SIGMOD, 2016, pp. 877–892.
[35] J. Wang, N. Tang, Towards dependable data repairing with fixing rules, in: Proc. of SIGMOD, 2014, pp. 457–468.
[36] X. Wang, Q.Z. Sheng, L. Yao, X. Li, X.S. Fang, X. Xu, B. Benatallah, Truth discovery via exploiting implications from multi-source data, in: Proc. of CIKM, 2016, pp. 861–870.
[37] H. Xiao, J. Gao, Q. Li, F. Ma, L. Su, Y. Feng, A. Zhang, Towards confidence in the truth: a bootstrapping based truth discovery approach, in: Proc. of SIGKDD, 2016, pp. 1935–1944.
[38] Y. Xu, W. Yin, A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion, SIAM J. Imaging Sci. 6 (3) (2013) 1758–1789.
[39] M. Yakout, A.K. Elmagarmid, J. Neville, M. Ouzzani, I.F. Ilyas, Guided data repair, PVLDB 4 (5) (2011) 279–289.
[40] C. Ye, H. Wang, T. Ma, J. Gao, H. Zhang, J. Li, AutoRepair: an automatic repairing approach over multi-source data, Knowl. Inf. Syst. (2018), doi:10.1007/s10115-018-1284-9.
[41] C. Ye, H. Wang, J. Li, H. Gao, S. Cheng, Crowdsourcing-enhanced missing values imputation based on Bayesian network, in: Proc. of DASFAA, 2016, pp. 67–81.
[42] C. Ye, H. Wang, T. Ma, J. Gao, H. Zhang, J. Li, PatternFinder: pattern discovery for truth discovery, Knowl.-Based Syst. 176 (2019) 97–109.
[43] X. Yin, J. Han, S.Y. Philip, Truth discovery with multiple conflicting information providers on the web, TKDE 20 (6) (2008) 796–808.
[44] H. Zhang, Q. Li, F. Ma, H. Xiao, Y. Li, J. Gao, L. Su, Influence-aware truth discovery, in: Proc. of CIKM, 2016, pp. 851–860.
[45] B. Zhao, J. Han, A probabilistic model for estimating real-valued truth from conflicting sources, in: Proc. of QDB, 2012.
[46] B. Zhao, B.I. Rubinstein, J. Gemmell, J. Han, A Bayesian approach to discovering truth from conflicting sources for data integration, PVLDB 5 (6) (2012) 550–561.
[47] Y. Zhou, J. He, Crowdsourcing via tensor augmentation and completion, in: Proc. of IJCAI, 2016, pp. 2435–2441.
[48] Y. Zhou, J. He, A randomized approach for crowdsourcing in the presence of multiple views, in: Proc. of ICDM, 2017, pp. 685–694.
[49] Y. Zhou, A.R. Nelakurthi, J. He, Unlearn what you have learned: adaptive crowd teaching with exponentially decayed memory learners, in: Proc. of SIGKDD, 2018, pp. 2817–2826.
[50] Y. Zhou, L. Ying, J. He, Multic^2: an optimization framework for learning from task and worker dual heterogeneity, in: Proc. of SIAM, 2017, pp. 579–587.