Enriching Data Imputation under Similarity Rule Constraints

Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, *Member, IEEE*, and Jianmin Wang

Abstract—Incomplete information often occurs along with many database applications, e.g., in data integration, data cleaning, or data exchange. The idea of data imputation is often to fill the missing data with the values of its neighbors who share the same/similar information. Such neighbors could either be identified certainly by editing rules or extensively by similarity relationships. Owing to data sparsity, the number of neighbors identified by editing rules w.r.t. value equality is rather limited, especially in the presence of data values with *variances*. To enrich the imputation candidates, a natural idea is to extensively consider the neighbors with similarity relationship. However, the candidates suggested by these (heterogenous) similarity neighbors may conflict with each other. In this paper, we propose to utilize the similarity rules with tolerance to small variations (instead of the aforesaid editing rules with strict equality constraints) to rule out the invalid candidates provided by similarity neighbors. To enrich the data imputation, i.e., imputing the missing values *more*, we study the problem of maximizing the missing data imputation. Our major contributions include (1) the NP-hardness analysis on solving as well as approximating the problem, (2) exact algorithms for tackling the problem, and (3) efficient approximation with performance guarantees. Experiments on real and synthetic data sets demonstrate the superiority of our proposal in filling accuracy. We also demonstrate that the record matching application is indeed improved, after applying the proposed imputation.

Index Terms—Similarity rules, similarity neighbors, data imputation

1 Introduction

Incomplete data (a.k.a. null or missing values) have been widely recognized as a typical data quality issue [1], owing to incomplete entry, inaccurate extraction or heterogeneous schemas, e.g., in Web autonomous databases [2]. The idea of imputing missing values is to explore the neighbors sharing same/similar information in the dataset. This imputation idea has been successfully applied in various areas such as analyzing the variance of planned experiments, survey sampling, multivariate analysis, and so on [3].

Example 1. Table 1 illustrates an example of incomplete data. For instance, the value of t_3 on attribute House Number is not available, i.e., a null cell denoted by $t_3[\text{HouseNumber}] = \text{'\_'}$.

An editing rule [4], $(((\text{Name}, \text{Name}), (\text{Street}, \text{Street})) \rightarrow (\text{HouseNumber}, \text{HouseNumber}), t_p = ())$, states that if two tuples t_i, t_j share equal Name and Street values, the missing $t_j[\text{HouseNumber}]$ can be filled by the non-null $t_i[\text{HouseNumber}]$.[1] As a neighbor of t_j, tuple t_i is identified certainly by the editing rule, given their exactly equal Name and Street values. Unfortunately, none of the tuples in Table 1 share the same Name and Street values with t_3, i.e., t_3 has no equality neighbor and cannot be filled.

To impute the missing value in t_3, the similarity-based method [5] finds k nearest neighbors that are similar to t_3 on complete attributes Name and Street. Without knowing which k value would lead to a better imputation for a particular incomplete tuple, the same k value is preset for imputing all the incomplete tuples, e.g., $k = 2$. In this case, not only the most similar t_6 but also the second similar t_4 will be considered as the neighbors of t_3. The values 86402 and #531 of similarity neighbors are thus considered as the candidates of imputing $t_3[\text{HouseNumber}]$. However, it cannot tell which one should be the correct imputation, 86402 or #531.

Consider a similarity rule in the form of differential dependencies (DD) [6].

$$\text{DD}_1 : (\text{Name}, \text{Street} \rightarrow \text{HouseNumber}, \langle [0, 1], [0, 7], [0, 3] \rangle)$$

It states that if two tuples have similar Name (i.e., having Name value distance[2] in the range of $[0, 1]$) and Street values (with distance in $[0, 7]$), they must share similar HouseNumber values as well (having HouseNumber value distance in $[0, 3]$).[3]

Referring to DD_1, tuples t_3 and t_6, sharing similar Name and Street values, must have similar HouseNumber values. However, the candidate 86402 suggested by the aforesaid similarity neighbors has distance to $t_6[\text{HouseNumber}]$ greater than 3, i.e., violating DD_1. 86402

1. t_i with non-null $t_i[\text{HouseNumber}]$ is regarded as reference data [4].

2. For example edit distance, see [7] for a survey of string similarity.

3. Similarity rules with distance thresholds can either be specified by domain experts or discovered from data [8].

- *S. Song, Y. Sun, A. Zhang, and J. Wang are with Tsinghua University, Beijing 100084, China. E-mail: {sxsong, jimwang}@tsinghua.edu.cn, {sy17, zaq13}@mails.tsinghua.edu.cn.*
- *L. Chen is with the Hong Kong University of Science and Technology, Hong Kong. E-mail: leichen@cse.ust.hk.*

Manuscript received 19 June 2018; revised 21 Oct. 2018; accepted 15 Nov. 2018. Date of publication 23 Nov. 2018; date of current version 8 Jan. 2020. (Corresponding author: Shaoxu Song.)
Recommended for acceptance by S. Sadiq.
Digital Object Identifier no. 10.1109/TKDE.2018.2883103

TABLE 1
Example Data with (Multi-)Incomplete Attributes

	Name	Street	House Number
t_1	Susan K Michel	–	–
t_2	Susan L Frank	JR	–
t_3	Terry Michel	JR	–
t_4	Susan Michel	JK Road	86402
t_5	Susan L Frank	Jordan Rd	#402
t_6	Terry K Michel	Jordan Rd	#531

is thus discarded, while #531 satisfying DD_1 is returned as the imputation of $t_3[\text{HouseNumber}]$.

(See Example 2 for imputing t_2. The imputation for multiple incomplete attributes in t_1 is shown in Example 4, where the incomplete attribute may appear in the left-hand-side of a rule.)

While editing rules [4] upon equality neighbors provide a limited number of imputations, the similarity neighbors [5] may return more but possibly self-contradictory imputation candidates. In this study, we propose to employ similar rules [6], which are defined on similarity rather than the strict equality and thus can be extensively utilized to rule out the enriched but irrational imputations.

1.1 Challenges

While similarity neighbors bring more imputation opportunities, it comes with new challenges. (1) Different from the certain fixes by equality neighbors with editing rules [4], multiple candidates may be suggested by similarity neighbors for imputing a cell. (2) The candidates for imputing two null cells could be *incompatible* w.r.t. similarity rules, owing to the complex similarity relationships[4] (see examples on t_2 and t_3 in Example 3). Such incompatibility is not considered in the certain fix by editing rules [4] either.

A practical question is thus whether all the null cells could be filled due to the incompatibility w.r.t. the similarity rules, namely *full filling* (Definition 3). If not, to what extend we can fill the null cells, known as *maximum filling* (Definition 2). Indeed, in light of the disability in imputing missing data (due to the aforesaid data sparsity and variety), it is naturally desirable to gain null-cell fillings as many as possible. As verified in the experiments in Section 4, the imputation performance could be significantly improved by gaining *more* fillings with the extensive similarity neighbors, compared to the equality-based barely-filled ones. Unfortunately, we find that maximizing the filling gain is Max-$\mathcal{SNP}$-hard, i.e., there exists $\varepsilon > 0$ such that achieving an approximation factor $(1 - \varepsilon)$ for the maximum filling problem is NP-hard (see Section 2 for details).

1.2 Contributions

The preliminary version of this paper [9] focuses on imputing single incomplete attribute, which is the right-hand-side (RHS) attribute of the given (DDs) rules. The imputation utilizes the complete left-hand-side (LHS) values to find neighbors and infer the missing RHS values referring to the (DDs) rules. This paper further extends the techniques to general cases for imputing multiple incomplete attributes including LHS attributes of the (DDs) rules. When LHS values are missing, we cannot rely on the DDs rules to find imputation candidates as in [9]. Our major contributions in this study are summarized as follows.

(1) We analyze the hardness of the similarity rule based imputation problem (Theorems 1 and 2) in Section 2.

(2) We present the algorithms for handling multiple incomplete attributes, where missing values may occur on the left-hand-side attributes of the (DDs) rules, in Section 3. The major techniques include the imputation candidate generation, an exact imputation algorithm via integer linear programming (ILP), its LP relaxation, and a randomized algorithm together with its derandomization. In particular, we show (in Propositions 4, 5, 9 and Theorem 8) that the approaches for handling multiple incomplete attributes are indeed equivalent to the methods in the preliminary version of this paper [9] when dealing with the special case of single incomplete attribute. Remarkably, a deterministic approximation factor is derived for the derandomization algorithm in Theorem 10.

(3) We report an extensive experimental evaluation on both effectiveness and efficiency, over real and synthetic data sets, in Section 4. It is highlighted that the imputation accuracy is improved by considering similarity neighbors under the constraints of similarity rules. We also demonstrate the improvement in the record matching application, after applying the proposed imputation.

Proofs of all the theoretical results can be found in the full version technique report [10].

2 Problem Statement

In this section, we formalize the problem of data imputation with similarity rules.

2.1 Preliminaries

Consider a relation I with schema $\mathcal{R}$. Let $\mathsf{dom}_I(A)$ denote all the values of an attribute A in I, i.e., $\mathsf{dom}_I(A) = \Pi_A(I)$.

2.1.1 Similarity Rules

For each attribute $A \in \mathcal{R}$, we associate a *similarity/distance metric*, d_A, which satisfies non-negativity, $d_A(a, b) \geq 0$; identity of indiscernibles, $d_A(a, b) = 0$ *iff* $a = b$; symmetry, $d_A(a, b) = d_A(b, a)$; where $a, b \in \mathsf{dom}_I(A)$ are values on attribute A. For example, the metric on a numerical attribute can be the absolute value of difference, i.e., $d_A(a, b) = |a - b|$. For a text attribute, we can adopt string similarity, e.g., *edit distance* (see [7] for a survey).

A *differential function* $\phi[A]$ on attribute A specifies a distance restriction by a range of metric distances over A. We say that two tuples t_1, t_2 in a relation I are *compatible* w.r.t. the differential function $\phi[A]$, denoted by $(t_1, t_2) \asymp \phi[A]$ or $(t_1[A], t_2[A]) \asymp \phi[A]$, if the metric distance of t_1 and t_2 on attribute A is within the range specified by $\phi[A]$, a.k.a. *satisfy/agree* with the distance restriction $\phi[A]$. As the metric is symmetric, it is equivalent to write $(t_2, t_1) \asymp \phi[A]$.

A differential function may also be specified on a set of attributes X, say $\phi[X]$, which denotes a pattern of differential functions (distance ranges) on all the attributes in X. We call $\phi[A]$ a *projection* on attribute A of $\phi[X], A \in X$.

A *differential dependency* (DD) [6] over $\mathcal{R}$ has a form $(X \rightarrow A, \phi[XA])$ where $X \subseteq R$ are determinant attributes, $A \in R$ is the dependent attribute, and $\phi[XA]$ is a differential

4. In essence, unlike equality, transitivity is not applicable to the similarity relationships, where similar (a, b) and similar (b, c) do not necessary imply similar (a, c).

TABLE 2
Example of Imputation Candidates

	...	Street	House Number
t_1	...	{Jordan Road, JK Road, –}	{86402, #531, –}
t_2	...	...	{86402, #402, –}
t_3	...	...	{86402, #531, –}

function on attributes X and A. It states that any two tuples from $\mathcal{R}$ satisfying the differential function $\phi[X]$ must satisfy $\phi[A]$ as well, $\phi[X]$ and $\phi[A]$ are the projections of differential function $\phi[XA]$ on X and A, respectively.

A relation I of $\mathcal{R}$ *satisfies* a DD, denoted by $I \models (X \rightarrow A, \phi[XA])$, if for any two tuples t_1 and t_2 in I, $(t_1, t_2) \asymp \phi[X]$ implies $(t_1, t_2) \asymp \phi[A]$, i.e., $(t_1, t_2) \models (X \rightarrow A, \phi[XA])$. We say a relation I satisfies a set Σ of DDs, $I \models \Sigma$, if I satisfies each DD in Σ.

2.1.2 Rule-Based Data Imputation

A *null* cell in a tuple $t_i \in I$ on attribute $A \in \mathcal{R}$ is denoted by $t_i[A] =\_$. It is regarded to be compatible with any other data w.r.t. distance restrictions, i.e., always having $(t_i, t_j) \asymp \phi[A]$. We consider an input with null cells such that $I \models \Sigma$.[5]

For each null cell $t_i[A] =\_$, we consider a set of imputation candidates $\mathsf{can}(t_i[A])$ from $\mathsf{dom}_I(A)$. It can be suggested by nearest neighbors (see Section 3.1 for details).

A *filling* I' of I is also an instance of $\mathcal{R}$ such that: (1) Existing non-null cells do not change, i.e., $t'_i[A] = t_i[A]$ if $t_i[A] \neq\_$, where $t'_i[A]$ is the cell in I' corresponding to $t_i[A]$ in I. (2) All the filled values come from the corresponding candidate set, i.e., $t'_i[A] \in \mathsf{can}(t_i[A])$ for each $t_i[A] =\_$. (3) Satisfaction of DDs is still retained, having $I' \models \Sigma$.

Example 2 (Example 1 continued). Consider another DD_2, in $\Sigma = \{\text{DD}_1, \text{DD}_2\}$, stating that two HouseNumber values a_1 and a_2 in the same Street should be similar (having $d_{\mathsf{HouseNumber}}(a_1, a_2) \leq 3$, with at most 3 different digits).

$$\text{DD}_2 : (\mathsf{Street} \rightarrow \mathsf{HouseNumber}, \langle [0,\ 0], [0,\ 3] \rangle).$$

The relation I in Table 1 satisfies this DD_2, since for any two tuples, e.g., t_5 and t_6, having the same Street (distance equal to 0 in the range of $[0, 0]$), it always has $(t_5, t_6) \asymp \phi[\mathsf{HouseNumber}]$, i.e., HouseNumber distance of t_5 and t_6 (equal to 3) is in $[0, 3]$.

Table 2 presents the sets of imputation candidates for null cells in Table 1, where the complete values are omitted by ...(see Example 4 for candidate generation).

I''_M in Table 3 presents a possible filling for the null cells in Table 1. Each filled value, e.g., $t'_2[\mathsf{HouseNumber}] =$ #402, comes from the candidate set in Table 2. The satisfaction w.r.t. Σ is retained, having $I''_M \models \Sigma$. In particular, the similarity rule DD_2 holds in the filled t'_2 and t'_3. That is, t'_2 and t'_3, with the same Street, must have similar filled HouseNumber values (#402 and #531 having distance within $[0, 3]$).

2.2 Problem Statement

As discussed in the introduction, referring to the difficulty in imputing a cell (owing to data sparsity, variety and heterogeneity), we target on the fillings that can fill more null cells and still satisfy the similarity rules.

5. For potential errors existing in non-null cells so that $I \not\models \Sigma$, a data repairing step [11] can be applied first on the non-null cells, which is out the scope of this study on imputing null cells.

TABLE 3
Example of Maximal and Maximum Fillings

I'_M	...	Street	House Number
t'_1	...	JK Road	86402
t'_2	...	...	86402
t'_3	...	...	–

I''_M	...	Street	House Number
t'_1	...	JK Road	86402
t'_2	...	...	#402
t'_3	...	...	#531

Consider a relation $I \models \Sigma$ with incomplete data. Let $I_M \subset I$ be the set of incomplete tuples with at least one missing value. We denote $\Delta(I'_M, I_M) = \{t'_i[A] \mid t'_i[A] \neq\_, t_i[A] =\_, t_i \in I_M\}$ the difference on cells between I_M and its filling I'_M w.r.t. Σ. We call the total number of cells filled in I'_M for I_M, $|\Delta(I'_M, I_M)|$, the *filling gain*.

Definition 1. *A filling I'_M is* maximal *if there does not exist any other filling I''_M of I_M, such that $\Delta(I'_M, I_M) \subset \Delta(I''_M, I_M)$.*

A result of '_' in a maximal fix I'_M denotes that the cell cannot be filled with the values from the candidate set. The reason is, as illustrated in the following example, filling candidates in different cells may conflict with each other.

Definition 2. *A filling I'_M is* maximum *if there is no other filling I''_M of I_M, such that $|\Delta(I'_M, I_M)| < |\Delta(I''_M, I_M)|$.*

Any maximum filling must be a maximal filling as well, according to the definitions.

Definition 3. *A filling I'_M is* full *if every null cell in I_M is filled.*

Example 3 (Example 2 continued). Consider $I_M = \{t_1, t_2, t_3\}$ with null cells in Table 1.

I'_M in Table 3 presents another filling of I_M. When $t'_2[\mathsf{HouseNumber}] =$ 86402, neither candidate 86402 nor #531 for $t_3[\mathsf{HouseNumber}]$ in Table 2 is compatible with the existing tuples or the aforesaid filled t'_2 w.r.t. Σ.

For DD_1, as presented in Example 1, candidate 86402 for $t_3[\mathsf{HouseNumber}]$ is not compatible with t_6.

For DD_2, since t_2 and t_3 have the same Street, they must have similar filled HouseNumber values. The other candidate #531 for $t_3[\mathsf{HouseNumber}]$, having distance 5 to $t'_2[\mathsf{HouseNumber}] =$ 86402, is not valid either.

Since there is no other candidate for consideration in $t_3[\mathsf{HouseNumber}]$, I'_M is already a maximal filling. However, I'_M is not maximum, given the full filling (also maximum and maximal) I''_M in Example 2 that fills all the null cells.

Since the filling candidates suggested by different similarity neighbors may conflict w.r.t. the constraints, obtaining the maximum filling is non-trivial.

Problem 1. *Given a relation $I \models \Sigma$ with incomplete data and the candidate set $\mathsf{can}(t_i[A])$ for each null cell, the* full filling

problem is to determine whether there is a full filling I'_M of I_M w.r.t. Σ.

Problem 2. *Given a relation $I \models \Sigma$ with incomplete data and the candidate set* $\mathsf{can}(t_i[A])$ *for each null cell, the* maximum filling *problem is to find a maximum filling I'_M of I_M w.r.t. Σ.*

The existence of a full filling can be determined by finding the maximum filling and checking whether all the null cells are filled. Unfortunately, both problems are hard.

Theorem 1. *The full filling problem is NP-complete.*

Referring to the hardness of determining full filling, it is not surprising to conclude the hardness of maximum filling. Indeed, if the maximum filling can be found (efficiently), it already determines whether a full filling exists or not.

Theorem 2. *The maximum filling problem is NP-hard.*

Recognizing the hardness of imputation w.r.t. DDs, we focus on approximation approaches below. Unfortunately, approximation of the imputation problem is also hard.

Theorem 3. *The maximum filling problem is Max-SNP-hard. That is, there exists an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation of the maximum filling is NP-hard.*

3 Imputing Multiple Incomplete Attributes

In this section, we devise algorithms for the imputation of missing values on multiple attributes under DDs Σ with various right-hand side attributes in $\mathcal{R}$ (in contrast to considering I_A with missing values on only one attribute A and a set of DDs Σ_A having the same right-hand side attribute A in Sections 3, 4, and 5 in the preliminary version of this paper [9]).

3.1 Candidate Generation

For multiple incomplete attributes, a tuple t_i may have missing values on the left-hand side attributes of a DD, and thus cannot rely on DDs to suggest imputation candidates. Instead, we employ complete tuples similar to t_i to suggest candidates, and use DDs to rule out invalid imputations.

3.1.1 Cell Candidates

For each tuple $t_i \in I_M$, we denote M_i the set of attributes having null values. Consider a distance/similarity metric on attributes $\mathcal{R} \setminus M_i$, $d_{\mathcal{R}\setminus M_i}$, for instance, by a simple summation of d_A on each complete attribute $A \in \mathcal{R} \setminus M_i$. We find a set K_i of k complete tuples t_c without null values from $I \setminus I_M$, which are most similar to t_i, a.k.a. k-nearest neighbors with the minimum distance $d_{\mathcal{R}\setminus M_i}(t_i, t_c)$.

The set of filling candidates for the null cell $t_i[A]$ is thus given by the values of k-nearest neighbors

$$\mathsf{can}(t_i[A]) = \{t_c[A] \mid t_c \in K_i\} \cup \{\_\}.$$

3.1.2 Tuple Candidates

Since there may have multiple attributes with missing values in a tuple t_i, we denote $\mathsf{can}(t_i)$ the tuple candidate set, by combining the cell candidates on each incomplete attribute of t_i.

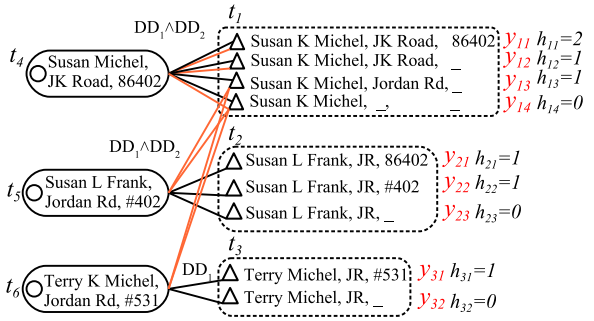


Fig. 1. Example of tuple candidates.

$$\mathsf{can}(t_i) = \Big\{ u \in \prod_{A \notin M_i} \{t_i[A]\} \times \prod_{A \in M_i} \mathsf{can}(t_i[A]) \mid (u, I \setminus I_M) \models \Sigma \Big\}. \tag{1}$$

Here, $(u, I \setminus I_M) \models \Sigma$ ensures that each tuple candidate u is not in violation to any complete tuple in I.

Example 4. Consider the imputation on multiple incomplete attributes in Table 1. We denote $I_M = \{t_1, t_2, t_3\}$ the tuples with missing values, and $M_1 = \{$Street, HouseNumber$\}$ the incomplete attributes in t_1.

Consider $\Sigma = \{\text{DD}_1, \text{DD}_2\}$ used in Example 3. It is notable that the incomplete attribute Street (in t_1) does not appear in the RHS attributes of all DDs.

For incomplete tuple t_1, tuples t_4, t_6 sharing similar Name value with t_1, are identified as k-nearest neighbors ($k = 2$) of tuple t_1. JK Road of t_4[Street] and Jordan Rd of t_6[Street] are thus suggested as two possible cell candidates to t_1[Street], i.e., $\mathsf{can}(t_1[\text{Street}])$ = {JK Road, Jordan Rd, _}. Similarly, 86402 and #531 are suggested to t_1[HouseNumber], having $\mathsf{can}(t_1[\text{HouseNumber}])$= {86402, #531, _}. There are 9 possible combinations that can be considered as tuple candidates $\mathsf{can}(t_1)$ for tuple t_1. However, according to the definition in Equation (1), the tuple candidates must be compatible with all the complete tuples.

Fig. 1 illustrates the tuple candidates of incomplete tuples in Table 1. Each triangle denotes a tuple candidate. The edge between a complete tuple and a tuple candidate denotes that some DDs apply. For instance, the first tuple candidate of t_1, (SusanKMichel, JK Road, 86402), and t_4 satisfy the LHS of DD_1 and DD_2, denoted by black and orange edges, respectively. With such DDs constraints, 4 tuple candidates for tuple t_1 will be reserved that can satisfy Σ w.r.t. complete tuples, as illustrated in Fig. 1.

Tuple candidates for other incomplete tuples, t_2 and t_3, can be obtained similarly referring to their own k-nearest neighbors and DDs.

3.2 Maximum Filling via ILP

Referring to the discipline of maximum filling in Section 2.2, we still expect to fill the null cells as many as possible. Let $y_{ij} = 1$ denote that the jth tuple candidate $u_{ij} \in \mathsf{can}(t_i)$ is selected to fill the incomplete tuple t_i, otherwise 0. Since multiple attributes may be filled in a tuple candidate, we introduce a weight h_{ij} for each tuple candidate $u_{ij} \in \mathsf{can}(t_i)$, to count how many null cells are filled, where $u[A] =_$ counts 0 in h_{ij}. In addition,

instead of counting 1 for each non-null cell fill, we can further weight the importance on each cell candidate $u[A] = a_l$, e.g., on how many times candidate $a_l \in \mathsf{can}(t_i[A])$ is suggested by k-nearest neighbors.

The maximum (weighted) filling problem w.r.t. tuple candidates is written as integer linear programming

$$\max \sum_{i=1}^{m} \sum_{j=1}^{g_i} y_{ij} \mathrm{h}_{ij} \tag{2}$$

$$\text{s.t.} \sum_{j=1}^{g_i} y_{ij} = 1, \qquad 1 \le i \le m \tag{3}$$

$$\mathrm{v}_{ijlk}(y_{ij} + y_{lk}) \le 1, \qquad 1 \le i \le m, 1 \le j \le g_i, \quad 1 \le l \le m, 1 \le k \le g_l \tag{4}$$

$$y_{ij} \in \{0, 1\}, \qquad 1 \le i \le m, 1 \le j \le g_i, \tag{5}$$

where $g_i = |\mathsf{can}(t_i)|$, and v_{ijlk} is a constant such that $\mathrm{v}_{ijlk} = 0$ if $(u_{ij}, u_{lk}) \models \Sigma, u_{ij} \in \mathsf{can}(t_i), u_{lk} \in \mathsf{can}(t_l)$, otherwise 1.

Referring to Equation (3), $\sum_{j=1}^{g_i} y_{ij} = 1$, there must be one and only one tuple candidate say $u_{ij} \in \mathsf{can}(t_i)$ selected for imputing the incomplete tuple t_i, i.e., $y_{ij} = 1$. The second constraint Equation (4) specifies that, if any two tuple candidates $u_{ij} \in \mathsf{can}(t_i), u_{lk} \in \mathsf{can}(t_l)$ are selected, i.e., $y_{ij} = y_{lk} = 1$, they must satisfy all the DDs, $(u_{ij}, u_{lk}) \models \Sigma$, i.e., $\mathrm{v}_{ijlk} = 0$, in order to meet the requirement of $\mathrm{v}_{ijlk}(y_{ij} + y_{lk}) \le 1$.

By formulating the maximum filling as ILP in Equations (2), (3), (4), (5), we can employ any existing ILP solver, e.g., [12]. Let m denote the number of incomplete tuples in I_M, g be the maximum number of tuple candidates for an incomplete tuple in I_M and sgm be the number of constraints in ILP. Referring to [12], time complexity is thus $O(2^{(1-\mathsf{poly}(1/s))gm})$.

We show that the ILP for single incomplete attribute in Section 3 in the preliminary version of this paper [9] is indeed a special case of the aforesaid ILP for multiple incomplete attributes.

Proposition 4. *When all the DDs in Σ have the same RHS attribute A, each incomplete tuple only has one single incomplete attribute A, and each non-null cell candidate counts one in h_{ij}, then the ILP in Equations (2), (3), (4), (5) for general cases is equivalent to the ILP in Equations (1), (2), (3), (4) for single incomplete attribute in Section 3 in the preliminary version of this paper [9].*

Example 5. According to Equation (1) in Section 3.1.2, each incomplete tuple $t_i \in I_M$ may own several tuple candidates, which are compatible with all existing tuples. However, the tuple candidates may not be compatible with each other among different incomplete tuples, i.e., they may be in violation to some tuple candidates of other incomplete tuples. For instance, as illustrated in Fig. 2, the tuple candidates (..., ..., 86402) for t_2 and (..., ..., #531) for t_3 are not compatible, owing to the distance restriction of DD_2. (We denote the complete attribute value by ... for simplicity.)

By formalizing the problem as integer linear programming, each tuple candidate is associated with a variable y_{ij}. For each pair of incompatible tuple candidates, e.g., the aforesaid first tuple candidates for t_2 and t_3, we put $y_{21} + y_{31} \le 1$. That is, these two tuple candidates cannot appear together in a solution with both $y_{21} = 1$ and $y_{31} = 1$.

To maximize $\sum_{i=1}^{m} \sum_{j=1}^{g_i} y_{ij} \mathrm{h}_{ij}$ in Equation (2), a possible ILP solution is $y_{11} = y_{22} = y_{31} = 1, y_{12} = y_{13} = y_{14} = y_{21} = y_{23} = y_{32} = 0$. It leads to a filling t'_1[Street] = JK Road, t'_1[HouseNumber]=86402, t'_2[HouseNumber] = #402, t'_3[HouseNumber] = #531.

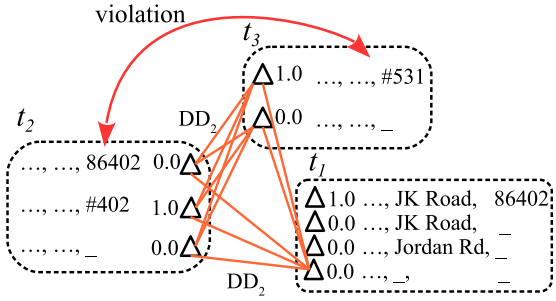


Fig. 2. Imputation by linear programming.

3.3 Approximation by LP Relaxation and Round

Referring to the hardness of approximation, we study heuristics for constructing feasible solutions by utilizing linear programming (LP) relaxation of ILP, i.e., change the constraint of $y_{ij} \in \{0, 1\}$ in Equation (5) to $0 \le y_{ij} \le 1$.

We first introduce the following subsumption relationship between tuple candidates (i.e., u_1 fills more than u_2).

Definition 4. *A tuple candidate u_1* subsumes *another u_2, denoted by $u_1 \succ u_2$, if (1) for each $u_2[A] \neq_-, A \in \mathcal{R}$, it has $u_1[A] = u_2[A]$, and (2) there exists an $A \in \mathcal{R}$ such that $u_1[A] \neq_-, u_2[A] =_-$.*

Algorithm 1. ROUND(I_M, Σ)

Input: I_M with tuple candidate sets and a set Σ of DDs
Output: A filling I'_M

```
1: let y be a solution of linear programming
2: for each t_i ∈ I_M do
3:   for each u_il ∈ can(t_i) do
4:     if u_il ⊁ t_i then
5:       set y_il to negative
6: I'_M := I_M
7: sort y_ij ∈ y in descending order of y_ij h_ij
8: while I'_M is not full do
9:   unchanged := TRUE
10:  for each y_ij > 0 do
11:    let u_ij be the j-th candidate in can(t_i)
12:    if (u_ij, I'_M) ⊨ Σ then
13:      t'_i := u_ij
14:      set y_il to negative for u_il ∈ can(t_i) with u_il ⊁ u_ij
15:      unchanged := FALSE
16:      break
17:  if unchanged is TRUE then
18:    return I'_M
```

Algorithm 1 presents the pseudocode of the ROUND algorithm for handling multiple incomplete attributes. Line 4 first eliminates those tuple candidates u_{il} that have no additional contribution to the current t_i, i.e., $u_{il} \not\succ t_i$, u_{il} cannot fill more over t_i. In each iteration, a

tuple candidate $u_{ij} \in \mathsf{can}(t_i)$ with the maximum $y_{ij}\mathrm{h}_{ij}$ and no violation to the other tuples w.r.t. Σ is assigned as t'_i. In particular, all the other candidates $u_{il} \in \mathsf{can}(t_i)$ that cannot fill more than u_{ij}, $u_{il} \not\succ u_{ij}$, could not further contribute to the filling and thus can be pruned (by setting y_{il} to negative), in Line 14.

Proposition 5. *ROUND Algorithm 1 always returns a maximal filling. When all the DDs in Σ have the same RHS attribute A, each incomplete tuple only has one single incomplete attribute A, and each non-null cell candidate counts one in h_{ij}, Algorithm 1 is equivalent to ROUND algorithm in the preliminary version of this paper [9] for handling single incomplete attribute.*

For a candidate u_{ij}, Line 12 takes $O(m)$ time to check whether it is compatible with the filled tuples in I'_M. In the worst case, Line 10 has to traverse mg tuple candidates to get a valid filling, where g is the maximum number of tuple candidates for a tuple in I_M. In each iteration of Line 8, at least one null cell is filled, i.e., having $O(m)$ iterations in worst case. The complexity of ROUND algorithm is $O(m^3 g)$.

Example 6 (Example 5 continued). Suppose that the numbers attached to each tuple candidate in Fig. 2 denote a LP solution. ROUND Algorithm 1 considers the tuple candidate with the highest $y_{ij}\mathrm{h}_{ij}$ in each iteration. Referring to the h_{ij} values given in Fig. 1, it first selects (..., JK Road, 86402) for t'_1, whose $y_{ij}\mathrm{h}_{ij}$ is the largest 2.0. The next largest $y_{ij}\mathrm{h}_{ij}$ is then considered, say $y_{22}\mathrm{h}_{22} = 1.0$, assigning (. . ., . . ., #402) to t'_2. And similarly, we have (. . ., . . ., #531) for t'_3. A full filling for incomplete data in Table 1 is generated.

3.4 Randomized Imputation

To enrich the imputation in terms of filling gain, we study the algorithm with performance guarantee on approximating the imputation over multiple incomplete attributes.

3.4.1 Probability

Since multiple null cells may be imputed by a tuple candidate, we take h_{ij} into consideration. Intuitively, the more the null cells are filled (larger h_{ij}), the higher the probability of the candidate being selected in randomized imputation is.

$$\Pr[t'_i = u_{ij}] = \begin{cases} \frac{y_{ij}\mathrm{h}_{ij}+\epsilon}{(g_i-1)\epsilon+1+\sum_{j=1}^{g_i} y_{ij}\mathrm{h}_{ij}}, \exists A \in M_i, u_{ij}[A] \neq_- \\ \frac{1}{(g_i-1)\epsilon+1+\sum_{j=1}^{g_i} y_{ij}\mathrm{h}_{ij}}, \forall A \in M_i, u_{ij}[A] =_- \end{cases} \quad (6)$$

$\epsilon \geq 0$ trades off the contribution of LP estimations. A larger ϵ denotes weaker effect by the LP solution y_{ij} (as evaluated in Section 4.4).

Considering all tuple candidates $u_{ij} \in \mathsf{can}(t_i)$, we have

$$\sum_{u_{ij} \in \mathsf{can}(t_i)} \Pr[t'_i = u_{ij}] = 1. \quad (7)$$

3.4.2 Random Algorithm

The candidate set $\mathsf{can}(t_i)$ in RANDOM Algorithm 2 considers all the possible tuple candidates u_{ij} for each incomplete tuple $t_i \in I_M$, with the probability $\Pr[t'_i = u_{ij}] > 0$.

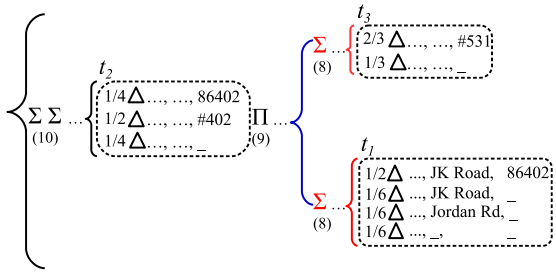


Fig. 3. Imputation with probability.

Proposition 6. *There exists a randomized algorithm which finds a full filling to any fully-fillable instance in expected time $\mathcal{O}((1/p)^m)$, where p is the minimum probability of a tuple candidate being selected.*

Algorithm 2. RANDOM(I_M, Σ)

Input: I_M with tuple candidate sets and a set Σ of DDs
Output: A filling I'_M

```
1: initialize probabilities
2: repeat
3:   for each t_i ∈ I_M do
4:     randomly draw a tuple candidate u_ij ∈ can(t_i) with
       probability Pr[t'_i = u_ij] for t'_i
5:     if t'_i = u_ij is in violation with any t'_l then
6:       t'_i[A] :=_, A ∈ M_i
7:   I'_M := ROUND(I'_M, Σ)
8:   rank I'_M in the top-k list K
9: until ℓ times
10: return the top-1 filling in K with the highest weighted filling gain
```

Obviously, all the possible maximal/maximum fillings could be generated under such a consideration.

Proposition 7. *All the possible maximal/maximum fillings could be generated by the RANDOM algorithm.*

A maximal filling is returned by calling the ROUND algorithm on I'_M in Line 7, referring to Proposition 5.

Example 7 (Example 6 continued). With the LP solution in Fig. 2 and the corresponding h_{ij} in Fig. 1, we obtain the probability of each tuple candidate according to Equation (6), denoted as the numbers attached to each triangle in Fig. 3 (with $\epsilon = 1$). Algorithm 2 randomly selects a tuple candidate with the probability for each incomplete tuple, e.g., t'_1 = (. . ., JK Road, 86402), t'_2 = (. . ., . . ., #402), t'_3 = (. . ., . . ., #531).

3.4.3 Expectation on Weighted Filling Gain $\mathbf{E}[H]$

Let random variable H denote the weighted filling gain, i.e., $\sum_{i=1}^{m}\sum_{j=1}^{g_i} y_{ij}\mathrm{h}_{ij}$ as in Equation (2). We study the expectation of the weighted filling gain, $\mathbf{E}[H]$, by the aforesaid randomized imputation.

Consider an assignment $t'_i = u_{ij}$, $u_{ij} \in \mathsf{can}(t_i)$, of any tuple $t_i \in I_M$. For any tuple t_l in I_M, the probability of $t'_i = u_{ij}$ compatible with t'_l is

$$\Pr[(t'_i = u_{ij}, t'_l) \models \Sigma] = \sum_{u_{lk} \in \mathsf{can}(t_l), (u_{ij}, u_{lk}) \models \Sigma} \Pr[t'_l = u_{lk}]. \quad (8)$$

The probability of $t'_i = u_{ij}$ compatible with all the tuples $t_l \in I_M$ is thus

$$\Pr\left[(t'_i = u_{ij}, I'_M) \models \Sigma\right] = \prod_{t_l \in I_M \setminus t_i} \Pr[(t'_i = u_{ij}, t'_l) \models \Sigma]. \quad (9)$$

Considering all the tuple candidates $u_{ij} \in \mathsf{can}(t_i)$ of each $t_i \in I_M$, with probability $\Pr[t'_i = u_{ij}]$, we have

$$\mathbf{E}[H] = \sum_{i=1}^{m} \sum_{j=1}^{g_i} \mathrm{h}_{ij} \Pr[t'_i = u_{ij}] \Pr\left[(t'_i = u_{ij}, I'_M) \models \Sigma\right]. \quad (10)$$

Example 8 (Example 7 continued). Given the tuple candidate probability of each incomplete tuple in Fig. 3, we show the computation of $\mathbf{E}[H]$ referring to Equations (8), (9), (10).

Consider t_2 as t_i and t_1 as t_l in Equation (8). The first tuple candidate $u_{21} = (\ldots, \ldots, \mathsf{86402})$ of t_2, considered as u_{ij}, is not in violation with any tuple candidate of t_1. The probability of $t'_2 = u_{21}$ compatible with t'_1 is $\Pr[(t'_2 = u_{21}, t'_1) \models \Sigma] = \frac{1}{2} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 1$.

For the other incomplete tuple t_3, the aforesaid $u_{21} \in \mathsf{can}(t_2)$ is in violation to the first candidate of t_3, i.e., $(\ldots, \ldots, \#531)$. Thereby, the probability of $t'_2 = u_{21}$ compatible with t'_3 is $\Pr[(t'_2 = u_{21}, t'_3) \models \Sigma] = \frac{1}{3}$.

By considering all the other incomplete tuples $\{t_1, t_3\}$ in Equation (9), we have the probability of $t'_2 = u_{21}$ compatible with them $\Pr\left[(t'_2 = u_{21}, I'_M) \models \Sigma\right] = 1 * \frac{1}{3} = \frac{1}{3}$.

For each tuple candidate of all incomplete tuples, we can compute such a probability, e.g., $\Pr\left[(t'_3 = (\ldots, \ldots, \#531), I'_M) \models \Sigma\right] = \frac{3}{4}$, $\Pr\left[(t'_1 = (\ldots, \mathsf{JK\ Road}, \mathsf{86402}), I'_M) \models \Sigma\right] = 1$, etc.

Finally, by the weighted filling gain definition in Equation (10), we have $\mathbf{E}[H] = 2 * \frac{1}{2} * 1 + 1 * \frac{1}{6} * 1 + 1 * \frac{1}{6} * 1 + 0 * \frac{1}{6} * 1 + 1 * \frac{1}{4} * \frac{1}{3} + 1 * \frac{1}{2} * 1 + 0 * \frac{1}{4} * 1 + 1 * \frac{2}{3} * \frac{3}{4} + 0 * \frac{1}{3} * 1 = \frac{29}{12}$.

Theorem 8. *RANDOM Algorithm 2 for multiple incomplete attributes returns a solution with the expected weighted filling gain* $\mathbf{E}[H] \geq \left(\frac{1}{(g-1)\epsilon+1+h}\right)^{m+1} OPT$, *where* OPT *is the optimal (maximum) weighted filling gain and* $h = \max_{ij} \mathrm{h}_{ij}$ *is the maximum weight of tuple candidate.*

We can show that the approximation bound in Theorem 8 in the preliminary version of this paper [9] is indeed a special case of Theorem 8 in this paper, when all the DDs in Σ have the same RHS attribute A, each incomplete tuple only has one single incomplete attribute A, each non-null cell candidate counts one in h_{ij}, and all the tuples in I_M are neighbors with each other. First, we have $h = 1$ since each candidate counts $\mathrm{h}_{ij} = 1$ with one single incomplete attribute A. If all the tuples in I_M are neighbors, we have $m = b$ (in Theorem 8 in [9]). Since _ is also counted in tuple candidates, it follows $g - 1 = c$ (in Theorem 8 in [9]).

3.5 Derandomization

RANDOM algorithm only has performance guarantee in the expectation of weighted filling gain. In this section, we present the DERAND algorithm for imputing multiple incomplete attributes, with a deterministic bound of approximation (Theorem 10).

3.5.1 Conditional Expectation

Let $\mathbf{E}[H \mid I'^{i-1}_M]$ be the conditional expectation, given a number of tuples $t_1, \ldots, t_{i-1}$ that have been filled, denoted as I'^{i-1}_M, i.e.,

$$\mathbf{E}[H \mid I'^{i-1}_M] = \mathbf{E}[H \mid t'_1 = u_1, \ldots, t'_{i-1} = u_{i-1}],$$

where $u_l \in \mathsf{can}(t_l)$, $l = 1, \ldots, i-1$, is the filled tuple of t_l. It denotes the total weighted filling gain of filled cells in $t_1, \ldots, t_{i-1}$ plus the expectation of weighted filling gain that can be obtained from the remaining $t_i, \ldots, t_m$. We have $\mathbf{E}[H \mid I'^{0}_M] = \mathbf{E}[H]$ initially, and $\mathbf{E}[H \mid I'^{m}_M]$ is the exact weighted filling gain of the filling I'^{m}_M.

Let $t'_i = u_{ij}, u_{ij} \in \mathsf{can}(t_i)$ be the next assignment. We study the incremental computation of the conditional expectation $\mathbf{E}[H \mid I'^{i}_M]$ from $\mathbf{E}[H \mid I'^{i-1}_M]$ by considering the following possible cases.

Case 1. If $t'_i = u_{ij}$ is in violation with any $t'_1 = u_1, \ldots, t'_{i-1} = u_{i-1}$, no valid solution can be generated, i.e., $\mathbf{E}[H \mid I'^{i}_M] = 0$.

Case 2. If $t'_i = u_{ij}$ is compatible with existing assignments, we further consider three sub-cases for updating the compatible probability in Equation (9) of remaining tuples $t_l, l = i+1, \ldots, m$, on candidate $u_{lk} \in \mathsf{can}(t_l)$.

Case 2.1. If $\forall (X \to A, \phi[XA]) \in \Sigma, (t'_i, t'_l = u_{lk}) \not\asymp \phi[X]$, the compatible probability will not change,

$$\Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M\right] = \Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i-1}_M\right].$$

Case 2.2. If $\exists (X \to A, \phi[XA]) \in \Sigma$ such that $(t'_i, t'_l = u_{lk}) \asymp \phi[X]$ and $(t'_i, t'_l = u_{lk}) \not\asymp \phi[A]$, we have

$$\Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M\right] = 0.$$

Case 2.3. If $\forall (X \to A, \phi[XA]) \in \Sigma, (t'_i, t'_l = u_{lk}) \asymp \phi[X]$ and $(t'_i, t'_l = u_{lk}) \asymp \phi[A]$, we have

$$\Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M\right] = \frac{\Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i-1}_M\right]}{\Pr\left[(t'_l = u_{lk}, t'_i) \models \Sigma \mid I'^{i-1}_M\right]}. \quad (11)$$

Once all $\Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M\right]$ are updated in the aforesaid cases, $\mathbf{E}[H \mid I'^{i}_M]$ is recomputed by Equation (10).

Example 9 (Example 8 continued). Suppose that we select $t'_2 = (\ldots, \ldots, \mathsf{86402})$ as the first assignment in Fig. 3. We illustrate how $\mathbf{E}[H \mid t'_2 = (\ldots, \ldots, \mathsf{86402})]$ is computed.

Since $t'_2 = (\ldots, \ldots, \mathsf{86402})$ is not compatible with $t'_3 = (\ldots, \ldots, \#531)$ in Fig. 2, i.e., Case 2.2, we have $\Pr\left[(t'_3 = (\ldots, \ldots, \#531), I'_M) \models \Sigma \mid t'_2 = (\ldots, \ldots, \mathsf{86402})\right] = 0$.

For a candidate $u_{lk} \in \mathsf{can}(t_l)$ compatible with t'_2, e.g., $\Pr\left[(t'_1 = (\ldots, \mathsf{JK\ Road}, \mathsf{86402}), I'_M) \models \Sigma\right] = 1$, we update it by dividing $\Pr[(t'_1 = (\ldots, \mathsf{JK\ Road}, \mathsf{86402}), t'_2) \models \Sigma] = 1$, i.e., Case 2.3. It has $\Pr\left[(t'_1 = (\ldots, \mathsf{JKRoad}, \mathsf{86402}), I'_M) \models \Sigma \mid t'_2 = (\ldots, \ldots, \mathsf{86402})\right] = 1$.

Finally, we have $\mathbf{E}[H \mid t'_2 = (\ldots, \ldots, \mathsf{86402})] = 2 * \frac{1}{2} * 1 + 1 * \frac{1}{6} * 1 + 1 * \frac{1}{6} * 1 + 0 * \frac{1}{6} * 1 + 1 + 1 * \frac{2}{3} * 0 + 0 * \frac{1}{3} * 1 = \frac{7}{3}$.

3.5.2 Imputation Guided by Conditional Expectation

Consequently, the DERAND algorithm chooses in each iteration an assignment that can maximize the expected

weighted filling gain of the remaining unassigned tuples.

$$t'_i = \arg\max_{u_{ij} \in \mathsf{can}(t_i)} \mathbf{E}[H \mid t'_1 = u_1, \ldots, t'_i = u_{ij}] \quad (12)$$

Algorithm 3. DERAND(I_M, Σ)

Input: I_M with tuple candidate sets and a set Σ of DDs
Output: A filling I'_M
1: initialize probabilities
2: **for** each $t_i \in I_M$ **do**
3: $E_{\max} := 0$
4: **for** each $u_{ij} \in \mathsf{can}(t_i)$ **do**
5: E : = CONEXP($\mathbf{E}[H \mid I_M^{i-1}], t_i, u_{ij}, \mathrm{h}_{ij}$)
6: **if** $E > E_{\max}$ **then**
7: $E_{\max} := E$
8: $t'_i := u_{ij}$
9: $\mathbf{E}[H \mid I'^{i}_M] := E_{\max}$
10: I'_M := ROUND(I'_M, Σ)
11: **return** I'_M
Procedure CONEXP($E, t_i, u_{ij}, \mathrm{h}_{ij}$)
Input: E denotes $\mathbf{E}[H \mid I_M^{i-1}]$ and assignment u_{ij} of t_i with confidence h_{ij}
Output: $\mathbf{E}[H \mid t'_1 = u_1, \ldots, t'_{i-1} = u_{i-1}, t'_i = u_{ij}]$
1: $E := E - \sum_k \mathrm{h}_{ik} \Pr[t'_i = u_{ik}] \Pr\left[(t'_i = u_{ik}, I'_M) \models \Sigma \mid I_M^{i-1}\right]$
2: $E := E + \mathrm{h}_{ij}$
3: **for** each $t_l \in I_M, l = i+1, \ldots, m$ **do**
4: **for** each $u_{lk} \in \mathsf{can}(t_l)$ **do**
5: $E := E - \mathrm{h}_{lk} \Pr[t'_l = u_{lk}] \Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I_M^{i-1}\right]$
6: **if** $(t'_i = u_{ij}, t'_l = u_{lk}) \models \Sigma$ **then**
7: update $\Pr[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M]$ by Equation (11)
8: $E := E + \mathrm{h}_{lk} \Pr[t'_l = u_{lk}] \Pr\left[(t'_l = u_{lk}, I'_M) \models \Sigma \mid I'^{i}_M\right]$
9: **return** E

Example 10 (Example 9 continued). Consider an assignment of t_2, e.g., $t'_2 = (\ldots, \ldots,$ #402). Line 5 in Algorithm 3 computes its $\mathbf{E}[H \mid t'_2 = (\ldots, \ldots,$ #402$)] = 2 * \frac{1}{2} * 1 + 1 * \frac{1}{6} * 1 + 1 * \frac{1}{6} * 1 + 0 * \frac{1}{6} * 1 + 1 + 1 * \frac{2}{3} * 1 + 0 * \frac{1}{3} * 1 = 3$.

For the other candidate, i.e., $t'_2 = (\ldots, \ldots, \_)$, we have $\mathbf{E}[H \mid t'_2 = (\ldots, \ldots, \_)] = 2 * \frac{1}{2} * 1 + 1 * \frac{1}{6} * 1 + 1 * \frac{1}{6} * 1 + 0 * \frac{1}{6} * 1 + 0 + 1 * \frac{2}{3} * 1 + 0 * \frac{1}{3} * 1 = 2$. As $\mathbf{E}[H \mid t'_2 = (\ldots, \ldots,$ #402$)] = 3$ is larger, $t'_2 = (\ldots, \ldots,$ #402$)$ is assigned.

For the next tuple t_3, we have

$$\mathbf{E}[H \mid t'_2 = (\ldots, \ldots, \text{\#402}), t'_3 = (\ldots, \ldots, \text{\#531})] = \frac{10}{3},$$

$$\mathbf{E}[H \mid t'_2 = (\ldots, \ldots, \text{\#402}), t'_3 = (\ldots, \ldots, \_)] = \frac{7}{3}.$$

As the first conditional expectation is larger, we select (..., ..., #531) as the assignment for t_3.

Finally, it leads to $\mathbf{E}[H \mid t'_1 = (\ldots, \text{JK Road}, 86402), t'_2 = (\ldots, \ldots, \text{\#402}), t'_3 = (\ldots, \ldots, \text{\#531})] = 4$, where all null cells are filled.

Proposition 9. *When all the DDs in Σ have the same RHS attribute A, each incomplete tuple only has one single incomplete attribute A, and each non-null cell candidate counts one in h_{ij}, then the DERAND algorithm for multiple incomplete attributes (Algorithm 3) is equivalent to Algorithm 3 in the preliminary version of this paper [9] for single incomplete attribute.*

3.5.3 Correctness and Performance Analysis

Finally, by showing that the maximum conditional expectation is non-decreasing, we can conclude that the final $\mathbf{E}[H \mid I'^{m}_M]$ returned by the DERAND algorithm is no less than the initial $\mathbf{E}[H \mid I'^{0}_M] = \mathbf{E}[H]$.

Theorem 10. *DERAND Algorithm 3 for multiple incomplete attributes guarantees to output a solution with weighted filling gain $\geq \mathbf{E}[H]$.*

The conditional expectation can be computed by considering all the other incomplete tuples except t_i and their corresponding tuple candidates with complexity $O(mg)$.

Considering all the m tuples in I_M and their tuple candidates, the complexity of the DERAND algorithm for multiple incomplete attributes (Algorithm 3) is $O(m^2g^2 + m^3g)$, where the last ROUND step in Line 10 takes $O(m^3g)$ time as analyzed at the end of Section 3.3.

4 EXPERIMENTS

This section reports the experiments on both effectiveness and efficiency of the proposed approaches. All programs are implemented in Java and the experiments were performed on a PC with 3.1 GHz CPU and 16 GB RAM.

4.1 Experimental Settings

We employ real and synthetic datasets, including the Restaurant[6] dataset with name, address, type and city information of 864 restaurants, the ASF dataset[7] consisting of 1.5 k tuples with 6 numerical attributes, and a synthetic dataset generated by the UIS database generator.[7]

Following the same line of evaluating data repairing techniques by artificially injecting errors [13], we randomly remove values from various attributes as missing data.

Let truth be the set of removed cell values and fill be the set of filling results returned by imputation algorithms. The accuracy is given by $accuracy = \frac{|\mathsf{truth} \cap \mathsf{fill}|}{|\mathsf{truth}|}$, i.e., the proportion of null cells that are accurately filled or recovered. If a null cell is failed to fill (leave '_' unchanged) or filled incorrectly, it will be counted negatively toward the accuracy as the wrong imputation. Moreover, to evaluate the imputation accuracy for numerical dataset, e.g., ASF dataset, we compare the imputed tuple t'_i to the corresponding truth $t_i^{\#}$, using the RMS (root-mean-square) error, i.e.,

$$\sqrt{\sum_{A \in \mathcal{R}} \frac{(t'_i[A] - t_i^{\#}[A])^2}{|\mathcal{R}|}}$$

Enlightened by discovering FDs from the complete part of incomplete data [2], DDs used in the experiments are also obtained by applying the discovery techniques [6], [8] (with manual verification, see the full version technique report [10] for discovery details and discovered rules).

4.2 Comparison with Existing Techniques

This experiment compares our proposed method with existing approaches. (1) Certain fixes are determined by using editing rules [4]. We consider all the complete tuples as reference data and discover FDs from the complete tuples (by [14]) as editing rules. (2) ERACER [15] iteratively learns statistical models and performs data imputation at the same time. The left-hand-side attributes of the aforesaid

6. http://www.cs.utexas.edu/users/ml/riddle/data.html
7. http://archive.ics.uci.edu/ml/datasets/

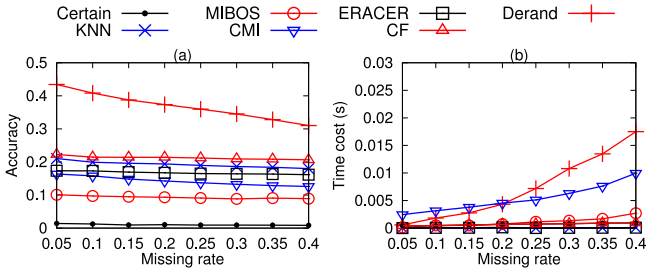


Fig. 4. Varying missing rates (Restaurant).

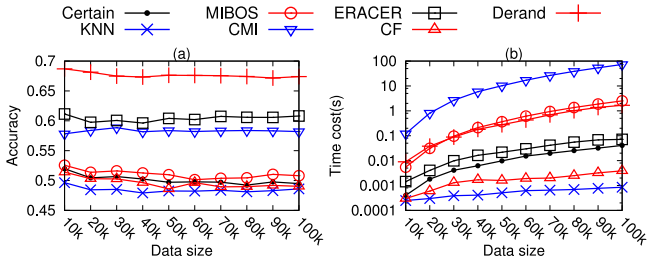


Fig. 5. Varying data sizes (UIS).

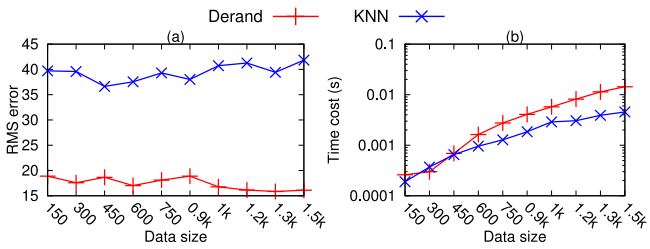


Fig. 6. Varying data sizes (ASF).

discovered FDs are utilized as the determinant attributes in the relational dependency network for ERACER. (3) MIBOS [16] considers the tuples similar to the incomplete tuple, where tuple similarity is defined on value equality. The complete tuple t_j having the maximum number of same values on complete attribute $\mathcal{R} \setminus M_i$ with the incomplete tuple t_i will be identified to impute the incomplete $t_i[A], A \in M_i$. It is worth noting that all the previous approaches are defined on the equality of attribute values, while our proposal further employs DD rules which can address similar attribute values. In order to conduct more fair comparison, we further study the methods with the consideration of attribute value similarity. (4) KNN [5] imputes incomplete tuples according to their k-nearest neighbors with the minimum distances over the complete attributes. We test various number k of neighbors, and report the one with best performance. (5) CMI [17] employs k-Means clustering to divide the dataset (including the instances with missing values) into clusters. The complete tuples in the same cluster will be used to impute the incomplete tuples. Again, we select the number of clusters having the best performance. To guarantee the fairness, the initial cluster centers are randomly assigned and the experiments are repeated 10 times. The mean value of imputation accuracy is returned to represent the performance for CMI. (6) CF [18] estimates prediction values by collecting preferences from collaborating users (analogous to tuples in imputation). We also consider the number of users (tuples) having the best performance for CF.

Fig. 4 reports the results of various missing rates. A missing rate, say 0.1, denotes that a cell has probability 0.1 of being a null cell. As shown, it is not surprising that the imputation accuracy drops with the increase of missing rate. (Detailed description of each method is given below.)

To evaluate scalability over larger data sizes, Fig. 5 reports the results on up to 100k tuples over UIS data. As shown, the imputation accuracy is generally stable. (See time cost explanation of Derand below.)

Certain [4], the constraint-based data cleaning method with editing rules, performs upon the equality relationships between tuples. With a limited number of equality neighbors, it is difficult to generate filling candidates and cannot lead to a unique fix when there are more than one candidate.

ERACER [15], the statistical-based approach over equal values, again shows lower imputation accuracy than our proposed Derand. The results verify our intuition on sparsity presented in the Introduction.

MIBOS [16] considers tuple similarity defined on value equality. With a very limited number of equal attribute values, the imputation is difficult. In contrast, our DDs-based Derand shows significantly higher accuracy owing to the successful identification of value similarity neighbors.

CMI [17] considers similarities over all the attributes, and thus has more opportunities in imputing missing values with higher accuracy (compared to MIBOS). However, strictly considering similarities over all the attributes, including those irrelevant ones, limits the power of finding similarity neighbors by CMI. Instead, by explicitly using only the (subset of) attributes specified in DDs, our Derand can identify and utilize more partially similar neighbors that do not belong to the same cluster. Thereby, the accuracy of Derand is higher. The CMI method also shows higher time cost over larger data sizes. The reason is that the k-means clustering algorithm in CMI converges more slowly over a large number of tuples (or with a higher duplication rate).

KNN [5] algorithm imputes incomplete tuples according to their k-nearest neighbors with the minimum distance over the complete attributes, rather than clusters in CMI. However, not all the candidates suggested by similarity neighbors are valid. By using DDs similarity rules, our proposed Derand approach can successfully rule out some irrational imputations referring to the constraints, and thus shows higher imputation accuracy in almost all the tests.

CF [18] makes predictions about the interests (analogous to missing values in the imputation problem) of a user (tuple) by collecting preferences from many collaborating users (tuples). It selects the users (tuples) which have similar preferences with the target user (incomplete tuple), and estimates the predictions based on the user (tuple) similarity, i.e., neighbors with higher similarity are more valuable for estimations. Unfortunately, similar to the KNN method, since the prediction values suggested by users (tuples) may conflict with each other owing to the heterogeneity issue, the imputation accuracy of CF is not high (close to KNN).

Fig. 6 evaluates the performance by varying the number of tuples over numerical data. As shown, our proposed method, selecting the best candidate, is more effective but takes more time than aggregating the k most similar candidates. The candidates suggested by k-nearest neighbors could be heterogeneous and conflict with the candidates of other tuples. Aggregating all these (potentially invalid) candidates to provide a filling for a missing cell is not accurate. On the contrary, our proposed Derand method can resolve

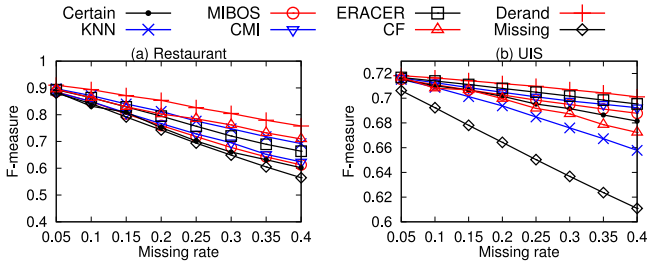


Fig. 7. Application in record matching (Restaurant & UIS).

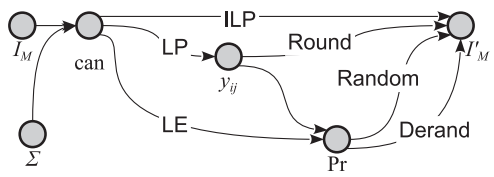


Fig. 8. Overview of approaches.

such conflict by considering the compatibility among candidates w.r.t. DDs. The selected best candidates thus have higher imputation accuracy. The corresponding time costs however are higher, which is not surprising referring to the hardness analyzed in Theorems 1-3.

4.3 Application in Record Matching

To further validate the effectiveness of applying imputation in real applications, we consider the accuracy of record matching application [19]. An existing rule-based record matching method [20] is directly implemented in the experiment, which utilizes matching dependencies (MDs) [21] to guide how records should be compared. It is performed over the Missing data without imputation, the imputed data by Certain [4], ERACER [15], MIBOS [16], CMI [17], KNN [5], CF [18], and our proposed Derand.

As shown in Fig. 7, the record matching accuracy (f-measure) is generally related to the filling accuracy in Fig. 4. Our Derand with higher imputation accuracy leads to better matching f-measure as well.

4.4 Performance of Proposed Techniques

This experiment evaluates the performance of different techniques in our proposal with various parameter settings. Fig. 8 provides an overview of all the approaches. Integer linear programming can be implemented by branch and bound, and linear programming employs simplex. When computing Pr by Equation (6), as introduced at the beginning of Section 3.4.1, for a large ϵ, each $u_{ij} \in \mathsf{can}(t_i)$ has a similar probability $\frac{1}{g_i - 1}$ of being selected. We consider particularly this large ϵ case (namely LE), as its Pr can be directly computed without calling the costly LP solver.

Fig. 9 reports the results over various ϵ which tunes the contribution of initial assignment y_{ij} in the probability of random selection in Equation (6). With the increase of ϵ, the contribution of y_{ij} by LP decreases, and the accuracy of Random(LP) drops in Fig. 9a. When ϵ is extremely large, Random(LP) shows almost the same result as Random (LE), which simply considers the equal probability. Round approach, not relying on the probability, does not change in this experiment. Remarkably, Derand(LE) approach shows the same result with Derand(LP) not only with large ϵ but also smaller ones. The results

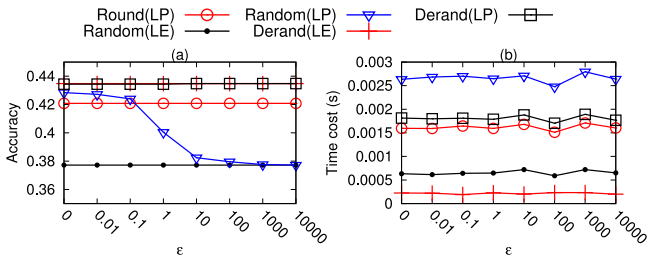


Fig. 9. Varying ϵ (Restaurant).

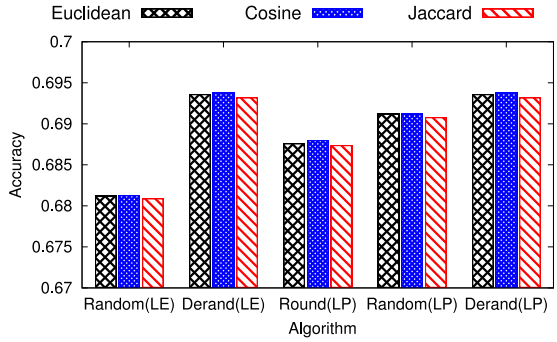


Fig. 10. Comparison on various similarity metrics (UIS).

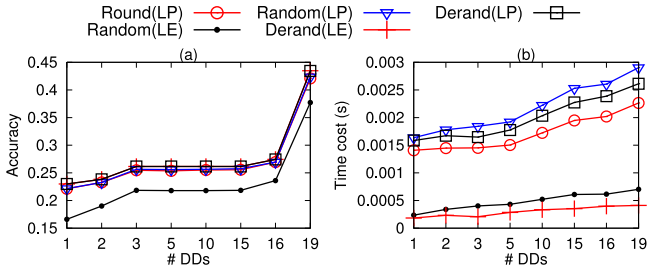


Fig. 11. Varying the number of used DDs (Restaurant).

demonstrate the robustness of Derand. In particular, since LE approaches simply assigning the same probability without calling the LP solver, the corresponding time costs are significantly lower. In this sense, Derand(LE) without calling the LP solver is preferred in practice, having lower time cost but higher accuracy.

Fig. 10 presents the results over UIS dataset where various similarity metrics are utilized in our proposed algorithms, i.e., Euclidean distance, cosine similarity, Jaccard coefficient [7]. As shown, the imputation methods show similar performance under different similarity functions. By default, for the similarity computing, we utilize 3-grams to divide each attribute value into a set of tokens and then compute the cosine similarity with tf-idf weighting.

To observe the accuracy of the proposed methods under different rules (presented in [10]), Fig. 11 reports the results on various numbers of DDs. In general, the imputation accuracy could be improved by given more reasonable rules. However, redundant semantics may exist among rules, which will not further improve the chance of imputation. In Fig. 11, given more than 3 DDs, the imputation accuracy cannot be further improved until the 16th DD is applied. In this sense, some DDs rules do affect the results more than others. The time cost increases when more DDs rules are given for examination. In practice, we can apply the existing techniques [6], [8] for reasoning about rules to eliminate redundant semantics, which is out the scope of this study.

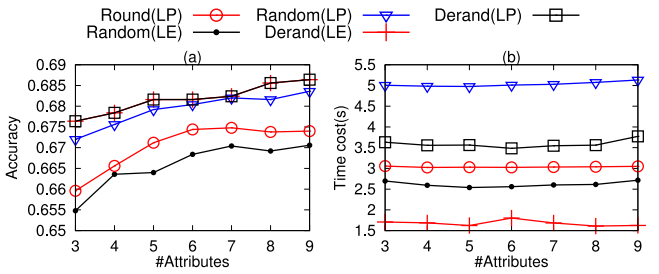


Fig. 12. Varying the number of attributes used for nearest neighbor computation in a tuple (UIS).

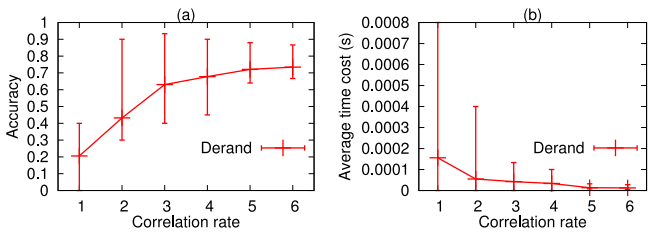


Fig. 13. Varying correlation rates (Restaurant).

Fig. 12 reports the results on various numbers of attributes used for nearest neighbor computation. We randomly select the attributes for a tuple according to the given number of attributes. The number of attributes, e.g., 3, denotes that there are three attributes randomly selected from the given schema for each tuple in nearest neighbor computation. With a moderate number of attributes, sufficient information can be utilized for nearest neighbor computation, and the selected nearest neighbors will be reliable. The imputation accuracy thus increases. However, with even higher dimensionality, finding appropriate neighbors is not trivial. By further increasing the number of attributes, the improvement on imputation accuracy is limited.

Fig. 13 observes the accuracy and average time cost of our proposed approach Derand with different levels of data correlation. Three points from higher to lower of each bar denote the maximum, average, minimum imputation accuracy and average time cost. The Restaurant dataset originally contains at most two tuples (versions) denoting one entity. To study various data distributions in terms of different levels of data correlation, we enrich more tuples for an entity by randomly choosing two alternative values (versions) on each attribute. As shown in Fig. 13a, with the increase of correlation rate, the imputation accuracy increases as well. It is not surprising since there are more reliable/correlated neighbors for candidate generation and more invalid candidates can be ruled out. Moreover, the corresponding average time cost of imputing an incomplete tuple drops in Fig. 13b, since more irrelevant candidates are filtered with higher correlation rates.

5 Related Work

The rule-based repairing, e.g., the certain fixes based on editing rules [4], could be applied to missing value imputation. However, as discussed in [9], the editing rules are built upon value equality relationships. Without sufficient neighbors, the null cells could barely be filled (i.e., lower imputation accuracy as observed in the experiments).

To find more neighbors for imputation, the similarity neighbor-based approaches [5], [16], [17] consider the tuples with high tuple-similarities defined over the complete attributes of the incomplete tuple. MIBOS [16] computes a tuple-similarity which is indeed defined on value equality, i.e., by counting the number of attributes with equal values. CMI [17] further considers clusters of most similar tuples, while KNN [5] employs only the k-nearest neighbors. The values of these similar tuples are aggregated as the imputation results of the incomplete tuple. It is worth noting that the suggested value candidates may conflict with each other. By applying the similarity rules, we can further rule out those irrational imputation candidates, and thus lead to more accurate imputation (in the experiments).

Statistical-based repairing [15], [22], [23] may support imputing missing values. Probabilistic correlations between reliable attributes with correct values and flexible attributes with dirty values are modeled in [22]. Repairing (and imputation) is thus to find the values that can maximize the likelihood. A complex relational dependency network is considered in [15] to model the probabilistic relationships among attributes. The cleaning process in [15] performs iteratively and terminates when the divergence of distributions is sufficiently small. As reported in [22], the approach [15] shows better performance in imputation. However, equality relationships among values are still considered in modeling the probabilities, without tolerance to small variances on values (the edit distance is only used to measure the similarity between a repair and its original value in [23] which is not applicable to imputation as aforesaid). Consequently, our Derand with extensive similarity neighbors show better imputation performance than the statistical-based [15].

It is also possible to utilize external sources of data for the imputation problem. Such methods match the incomplete tuples with the external information in master data [4], domain experts [24] and knowledge bases [25], for imputing null cells. [4] employs the matches between master data and the incomplete tuples, under the guidance of editing rules. In order to resolve ambiguity, expert feedback is involved in the cleaning process to confirm the results suggested by constraint-based cleaning techniques [24]. [25] interprets table semantics, maps it to the knowledge base, and generates imputation values from the knowledge base. In general, these methods heavily rely on the quality of external resources, which are usually expensive to employ for many domains. Since this study considers only the information available inside the dataset, we omit the unfair comparison to the methods with external sources in the experiments.

6 Conclusions

Imputing missing values of a tuple replies on others (neighbors) sharing the same/similar information. While candidates suggested by various similarity neighbors could conflict with each other due to heterogeneity, we propose to utilize similarity rules to rule out invalid candidates. In this paper, we (1) analyze the hardness of computing the maximum/maximal fillings (Theorems 1 and 2) as well as its approximation (Theorem 3); (2) present an exact algorithm by ILP and its LP relaxation; and (3) devise a randomized algorithm together with derandomization. In particular, the efficient approximation algorithms are devised with certain performance guarantees (Theorem 10). Experiments on real and synthetic data demonstrate that (1) our proposed methods such as Derand(LE) show higher imputation accuracy

and scale well in large data sizes; and (2) the accuracy of record matching application can be improved if we apply our proposal to fill the missing data first.

In this study, we consider only the accurate DD rules as constraints, to rule out the imputation candidates that do not satisfy the given constraints. It is promising to further employ imprecise DDs with the consideration of rule confidence. The trade-off between the accuracy of imputation results and getting more null cell filled seems non-trivial. For instance, a candidate a_1 may satisfy DD_1 with higher confidence but violate DD_2 with lower confidence. However, the imputation of tuple t_1 by candidate a_1 may prevent the imputation of another tuple t_2. On the other hand, the imputation of t_1 by another candidate a_2 violates the high confidence DD_1 but satisfies the low confidence DD_2. Nevertheless, it does not affect imputing t_2, i.e., both t_1 and t_2 can be filled. In this sense, advanced investigation is necessary on how to balance the rule accuracy and imputation gain. We leave this complicated scenario as the future study.

Acknowledgments

This work is supported in part by the National Key Research Program of China under Grant 2016YFB1001101 and China NSFC under Grants 61572272 and 71690231.

References

[1] D. Firmani, M. Mecella, M. Scannapieco, and C. Batini, "On the meaningfulness of "big data quality" (invited paper)," *Data Sci. Eng.*, vol. 1, no. 1, pp. 6–20, 2016. [Online]. Available: https://doi.org/10.1007/s41019-015-0004-7

[2] G. Wolf, H. Khatri, B. Chokshi, J. Fan, Y. Chen, and S. Kambhampati, "Query processing over incomplete autonomous databases," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 651–662.

[3] R. J. A. Little and D. B. Rubin, *Statistical Analysis with Missing Data*. New York, NY, USA: Wiley, 2002.

[4] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu, "Towards certain fixes with editing rules and master data," *Proc. VLDB Endowment*, vol. 3, no. 1, pp. 173–184, 2010. [Online]. Available: http://www.comp.nus.edu.sg/vldb2010/proceedings/files/papers/R15.pdf

[5] C. Huang and H. Lee, "A grey-based nearest neighbor approach for missing attribute value prediction," *Appl. Intell.*, vol. 20, no. 3, pp. 239–252, 2004. [Online]. Available: https://doi.org/10.1023/B:APIN.0000021416.41043.0f

[6] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM Trans. Database Syst.*, vol. 36, no. 3, 2011, Art. no. 16.

[7] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001. [Online]. Available: http://doi.acm.org/10.1145/375360.375365

[8] S. Song, L. Chen, and H. Cheng, "Parameter-free determination of distance thresholds for metric distance constraints," in *Proc. Int. Conf. Data Eng.*, 2012, pp. 846–857. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2012.46

[9] S. Song, A. Zhang, L. Chen, and J. Wang, "Enriching data imputation with extensive similarity neighbors," *Proc. VLDB Endowment*, vol. 8, no. 11, pp. 1286–1297, 2015. [Online]. Available: http://www.vldb.org/pvldb/vol8/p1286-song.pdf

[10] "Full version," [Online]. Available: http://ise.thss.tsinghua.edu.cn/sxsong/doc/imputation.pdf

[11] X. Chu, I. F. Ilyas, and P. Papotti, "Holistic data cleaning: Putting violations into context," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 458–469.

[12] R. Impagliazzo, S. Lovett, R. Paturi, and S. Schneider, "0-1 integer linear programming with a linear number of constraints," *Electron. Colloquium Comput. Complexity*, vol. 21, 2014, Art. no. 24. [Online]. Available: http://eccc.hpi-web.de/report/2014/024

[13] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro, "Messing up with BART: Error generation for evaluating data-cleaning algorithms," *Proc. VLDB Endowment*, vol. 9, no. 2, pp. 36–47, 2015. [Online]. Available: http://www.vldb.org/pvldb/vol9/p36-arocena.pdf

[14] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *Comput. J.*, vol. 42, no. 2, pp. 100–111, 1999.

[15] C. Mayfield, J. Neville, and S. Prabhakar, "ERACER: A database approach for statistical inference and data cleaning," in *Proc. ACM SIGMOD Conf.*, 2010, pp. 75–86. [Online]. Available: http://doi.acm.org/10.1145/1807167.1807178

[16] S. Wu, X. Feng, Y. Han, and Q. Wang, "Missing categorical data imputation approach based on similarity," in *Proc. IEEE Int. Conf. Syst. Man. Cybern.*, 2012, pp. 2827–2832. [Online]. Available: http://dx.doi.org/10.1109/ICSMC.2012.6378177

[17] S. Zhang, J. Zhang, X. Zhu, Y. Qin, and C. Zhang, "Missing value imputation based on data clustering," *Trans. Comput. Sci.*, vol. 1, pp. 128–138, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-79299-4_7

[18] R. M. Rodríguez, L. Martínez-López, D. Ruan, and J. Liu, "Using collaborative filtering for dealing with missing values in nuclear safeguards evaluation," *Int. J. Uncertainty Fuzziness Knowl.-Based Syst.*, vol. 18, no. 4, pp. 431–449, 2010. [Online]. Available: https://doi.org/10.1142/S0218488510006635

[19] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 1, pp. 1–16, Jan. 2007.

[20] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1995, pp. 127–138. [Online]. Available: http://doi.acm.org/10.1145/223784.223807

[21] W. Fan, J. Li, X. Jia, and S. Ma, "Reasoning about record matching rules," *Proc. VLDB Endowment*, 2009.

[22] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid, "Don't be scared: Use scalable automatic repairing with maximal likelihood and bounded changes," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 553–564. [Online]. Available: http://doi.acm.org/10.1145/2463676.2463706

[23] S. De, Y. Hu, V. V. Meduri, Y. Chen, and S. Kambhampati, "Bayeswipe: A scalable probabilistic framework for improving data quality," *J. Data Inf. Quality*, vol. 8, no. 1, pp. 5:1–5:30, 2016. [Online]. Available: http://doi.acm.org/10.1145/2992787

[24] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas, "Guided data repair," *Proc. VLDB Endowment*, vol. 4, no. 5, pp. 279–289, 2011. [Online]. Available: http://portal.acm.org/citation.cfm?id=1952378&CFID=12591584&CFTOKEN=15173685

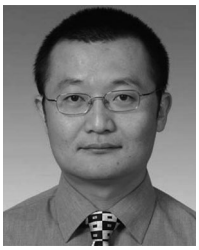
[25] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1247–1261. [Online]. Available: http://doi.acm.org/10.1145/2723372.2749431

Shaoxu Song is an associate professor with the School of Software, Tsinghua University, Beijing, China. His research interests include data quality and complex event processing. He has published more than 20 papers in top conferences and journals such as SIGMOD, VLDB, ICDE, the *ACM Transactions on Database Systems*, the *IEEE Transactions on Knowledge and Data Engineering*, *The VLDB Journal*, etc.

Yu Sun is working toward the PhD degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include data quality and data cleaning.

Aoqian Zhang is a working PhD degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include time series data cleaning and data imputation.

Jianmin Wang is a professor with the School of Software, Tsinghua University. His current research interests include unstructured data management, workflow and BPM technology, benchmark for database system, information system security, and large-scale data analytics.



Lei Chen is currently a professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include crowdsourcing, social networks, probabilistic and uncertain data, cloud data processing, and graph data. He is a member of the ACM, the IEEE, and the IEEE Computer Society.