Win-Win: On Simultaneous Clustering and Imputing over Incomplete Data

Yu Sun
College of Computer Science, Nankai University
sunyu@nankai.edu.cn

Jingyu Zhu
Nankai University
2013216@mail.nankai.edu.cn

Xiao Xu
Ant Group
suyan.xx@antgroup.com

Xian Xu
Ant Group
xx425168@antgroup.com

Yuyao Sun
Ant Group
sunyuyao.syy@antgroup.com

Shaoxu Song
Tsinghua University
sxsong@tsinghua.edu.cn

Xiang Li
Ping An Health Technology
lixiang453@pingan.com.cn

Xiaojie Yuan
Nankai University
yuanxj@nankai.edu.cn

ABSTRACT

Although clustering methods have shown promising performance in various applications, they cannot effectively handle incomplete data. Existing studies often impute missing values first before clustering analysis and conduct these two processes separately. However, inaccurate imputation does not necessarily contribute positively to the subsequent clustering. Intuitively, accurate imputation and clustering can serve and benefit from each other, where clustering-based imputation methods typically utilize cluster signals to impute incomplete data and accurate fillings are expected to bring more valuable data for clustering. Therefore, in this manuscript, rather than considering two tasks independently or conducting them respectively, we study simultaneous clustering and imputing over incomplete data. The immediate benefit is that such a strategy improves both clustering and imputation performance simultaneously, to get a win-win result. Our major technical highlights include (1) the problem formalization and NP-hardness analysis on computing simultaneous clustering and imputing results, (2) exact solutions by transforming the problem as the integer linear programming (ILP) formulation, and (3) efficient approximation algorithms based on the linear programming (LP) relaxation and local neighbors (LN) solution, with approximation guarantees. Experiments on various real-world datasets demonstrate the superiority of our work in clustering and imputing incomplete data.

PVLDB Reference Format:
Yu Sun, Jingyu Zhu, Xiao Xu, Xian Xu, Yuyao Sun, Shaoxu Song, Xiang Li, and Xiaojie Yuan. Win-Win: On Simultaneous Clustering and Imputing over Incomplete Data. PVLDB, 17(11): 3045 - 3057, 2024.
doi:10.14778/3681954.3681982

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/erssss/impute-SCI.

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:10.14778/3681954.3681982

Shaoxu Song (https://sxsong.github.io/) is the corresponding author.

1 INTRODUCTION

Data clustering is a fundamental and important technique that has a broad range of applications, including pattern recognition [41], bioinformatics [51], data compression [72], anomaly detection [25], etc. Meanwhile, due to sensor failure or network outage [49], incomplete data (a.k.a. null or missing values) are commonly observed, which would encumber both clustering results and corresponding applications as aforesaid [30, 33, 54, 75].

Effectively handling missing data during clustering analytics has thus become a crucial problem. A natural idea is the two-step method, which imputes incomplete data first and then applies clustering methods. Data scientists spend a lot of time designing imputation approaches based on various signals, e.g., constraints [55, 61], neighbors [19, 31], clusters [53, 54, 71, 74], statistics [52, 70].

However, as also shown in our empirical study (in Section 5.2), imputation methods cannot necessarily bring a positive impact on the clustering performance [29]. The reason is that the imputation objective is not highly correlated with the clustering target, resulting in the inaccurate imputation cannot serve the clustering performance. For instance, the constraint-based method [61] usually finds the maximum imputation result satisfying the constraints. Although clustering-based methods [53, 54, 71, 74] divide the incomplete dataset into clusters first and then use complete tuples in the same cluster to impute incomplete tuples, they do not consider the influence on the clustering performance either.

Our preliminary studies [58, 59] show that simultaneous clustering and cleaning can improve both data quality and clustering performance, where two tasks can serve and benefit from each other. Following the same line, simultaneous clustering and imputing incomplete data is a promising strategy, by taking full advantage of their mutual improvement [38]. That is, more accurate fillings are expected to bring more valuable data for clustering, benefiting downstream applications.[1] On the other hand, following the same intuition of clustering-based imputation methods, better clustering results provide more reliable evidence to guide data imputation as well.[2] Recent studies [30, 33, 54, 75] thus integrate imputation and clustering tasks into a unified framework, where two processes are conducted in turn. Unfortunately, they still do not construct a clear

[1] Please see an experimental application study in Section 5.5.
[2] Such a claim assumes the clustering correlation between missing data and complete data. Please see Section 2.2 for more detailed discussions.

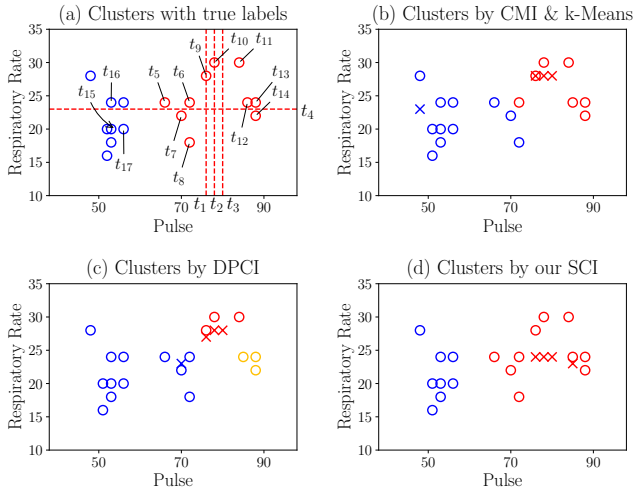

Figure 1: Horse [14] data over Pulse and Respiratory Rate attributes with real-world missing values on the attribute Respiratory Rate in tuples t_1, t_2, t_3 and on the Pulse attribute in tuple t_4. In subfigure (a) with true cluster labels, red points form the surgical lesion cluster, and blue points represent the non-surgical lesion cluster.

connection between imputation and clustering performance, where two processes are conducted respectively without making the best of their mutual benefits.

To ensure the successful improvement between clustering and imputing results, we study Simultaneous Clustering and Imputing incomplete data (SCI). Specifically, rather than treating imputation and clustering tasks independently or conducting them respectively, we consider and process two tasks simultaneously.[3] That is, the impact on the clustering performance is considered during data imputation, and vice versa. The immediate benefit is that our study could optimize both clustering and imputation results at the same time, taking full advantage of their mutual benefits.

Example 1. *Consider the practical application of Horse [14] for determining surgical lesions in horses. Two critical indicators, i.e., pulse and respiratory rate, as illustrated in Figure 1, are important for the disease diagnosis of horses. Unfortunately, owing to various reasons, there are real-world missing values in t_1[Respiratory Rate], t_2[Respiratory Rate], t_3[Respiratory Rate], and t_4[Pulse], which are denoted by dotted lines in Figure 1(a).*

CMI [74], as a clustering-based imputation method, initially partitions the incomplete instance into two clusters, depicted in blue and red respectively in Figure 1(b). The clustering process is conducted over the whole instance, including both complete tuples and incomplete tuples, according to their tuple distances on the complete attribute values. Then incomplete tuples t_1, t_2 and t_3 are imputed by those tuples in the same (red) cluster with them and t_4 is filled by complete tuples in the other (blue) cluster. Afterwards, k-Means clustering is applied to the imputation results. Unfortunately, as shown in Figure 1(b), the imputed tuples by CMI are mis-clustered, along with the inaccurate imputation results.

[3]Please see the formal problem definition in Section 2.2.

Regarding the unified framework DPCI [33], incomplete tuples are first imputed using 2-nearest-neighbors based on their complete attribute values, such as $\{t_{12}, t_{16}\}$ for t_4. As illustrated in Figure 1(c), the original separation of the surgical lesion cluster caused by missing values makes imputation values closely aligned with only one of the three separated clusters (marked in blue, red, and yellow, respectively), resulting in inaccurate fillings. For instance, since t_{12} and t_{16} belong to different clusters, the corresponding imputation result t'_4[Pulse] = 70 is thus inaccurate due to the aggregation of heterogeneous values from these separate clusters. Furthermore, training a clustering model over complete tuples, where t_{12}, t_{13}, and t_{14} form an individual yellow-marked cluster due to being far away from the others, also leads to inaccurate clustering results. That is, although DPCI integrates clustering and imputation tasks into one unified framework, conducting them separately is still hard to get accurate results.

Conversely, our SCI evaluates the performance of clustering and imputing tasks simultaneously, considering the potential fillings of all the missing values. As shown in Figure 1(d), we observe that the imputed values t'_1[Respiratory Rate] = 24, t'_2[Respiratory Rate] = 24, t'_3[Respiratory Rate] = 24, and t'_4[Pulse] = 86 can reconnect the surgical lesion cluster previously segmented due to missing values. Such a solution leads to improved results for clustering and imputation tasks, making it the preferred solution by SCI.

1.1 Contributions

Our major contributions in this study are summarized as follows.

(1) We formalize the optimal simultaneous clustering and imputing incomplete data problem and analyze its hardness (Lemma 1) in Section 2.
(2) We compute exact solutions by transforming the problem into the ILP formulation in Section 3.
(3) We design an approximate algorithm based on the LP relaxation in Section 4.2, with the polynomial-time complexity (Proposition 3) and approximation guarantees (Proposition 4).
(4) We further advance the approximation algorithm to be more efficient by considering LN solutions without calling LP solvers (Proposition 5) in Section 4.3, whose approximation performance is also ensured under the certain case (Proposition 6).

Finally, we report an extensive experimental evaluation in Section 5, compared with sixteen competitive baselines, including imputation methods, missing data tolerant clustering approaches and unified frameworks. Experiment results show the superiority of our work in improving both clustering and imputing performance. In addition, the application study on the downstream anomaly detection demonstrates the practicality of our methods.

2 PROBLEM STATEMENT

In this section, we provide the terminology used in this manuscript and formalize the problem of simultaneous clustering and imputing incomplete data, with the hardness analysis.

2.1 Preliminaries

Consider a relational instance $r = \{t_1, \ldots, t_n\}$ over schema $R = (A_1, \ldots, A_m)$. Each tuple $t_i \in r$ consists of cells $\{t_i[A_1], t_i[A_2], , \ldots, t_i[A_m]\}$, where $t_i[A_j]$ is the value of attribute A_j in tuple t_i.

2.1.1 Data Clustering. For each attribute $A_j \in R$, let Δ be any distance metric, having $0 \le \Delta(t_i[A_j], t_l[A_j]) \le 1$. For instance, it can be a min-max normalization distance [36] for numerical values, or a normalized Levenshtein distance [47] for string values.

By aggregating the distance on each attribute $A_j \in R$, we can obtain the tuple distance $\Delta(t_i[R], t_l[R])$, or simply $\Delta(t_i, t_l)$. L^2 norm, the Euclidean distance [36], is a commonly used distance metric, which serves as the default function to compute the tuple distance.[4]

To evaluate the clustering performance, for a set of centers $C \subset r$, we define the distance between each tuple $t_i \in r$ and C as the minimum tuple distance from t_i to any center $t_l \in C$,

$$\Delta(t_i, C) = \min_{t_l \in C} \Delta(t_i, t_l). \tag{1}$$

Since different clustering methods own different characteristics and objectives, it is hard to design a common objective effective for all kinds of them. However, the main idea of different clustering methods is generally similar, i.e., dividing close tuples into the same cluster. Enlightened by this, to serve more general scenarios, we follow the same objective with the most commonly used partition-based clustering [66]. Then the clustering objective for an instance r w.r.t. the centers $C \subset r$ can be calculated by summarizing the distance cost $\Delta(t_i, C)$ for each $t_i \in r$.

Definition 1 (Distance Cost). *Given a relational instance r and a set of cluster centers $C \subset r$, the* distance cost $\mathcal{D}(r, C)$ *is*

$$\mathcal{D}(r, C) = \sum_{t_i \in r} \Delta(t_i, C). \tag{2}$$

For an instance r, the goal of the clustering problem is to find at most κ tuples $C \subset r$ as cluster centers, such that the *distance cost* $\mathcal{D}(r, C)$ is minimized [66]. As discussed above, although such an objective cannot be strictly same as all the clustering methods, it requires all the tuples in each cluster similar to each other, i.e., conforming to the core idea of different clustering methods.

Example 2. *Consider the complete version of the relational instance r in Figure 1(b), with two cluster centers. According to Formula 1, the distance cost of t_{14} w.r.t. the cluster centers $C = \{t'_2, t_{17}\}$ is $\Delta(t_{14}, C) = \Delta(t_{14}, t'_2) = 0.81$, where t'_2[Respiratory Rate] $= 28$. Then the distance cost of instance r is calculated according to Formula 2, having $\mathcal{D}(r, C) = 0.81 + 0.61 + 0.30 + 0.16 + 0.0 + \cdots = 5.35$.*

2.1.2 Imputation Candidates. Let $r_M \subset r$ represent the set of incomplete tuples containing at least one missing value. For each tuple $t_i \in r_M$, we denote M_i as the set of attributes having null values in t_i.

Following the common procedure of existing data cleaning methods [55, 61, 70], for each null cell $t_i[A_j] = \perp$, we consider a set of *cell candidates* $\mathsf{can}(t_i[A_j])$. It can be simply the domain values $\mathsf{dom}(A_j) = \Pi_{A_j}(r)$ of attribute A_j, or narrowed down by existing imputation methods, e.g., suggested by k-nearest neighbors on complete attributes [19].[5] That is, we apply the assumption that the true value of each null cell $t_i[A_j] = \perp$ is included in the candidate set, which is a widely used assumption in data cleaning [37, 42].

[4]It is notable that any L^p metric is applicable in our work, depending on the specific user requirement.

[5]Experiments in Section 5.2 demonstrate that our methods show superiority against various baselines, simply with such kNN candidates.

Moreover, for each incomplete tuple $t_i \in r_M$, there may exist multiple attributes having missing values, i.e., $|M_i| > 1$. We thus consider *tuple candidates* $\mathsf{can}(t_i)$, by combining all the cell candidates on each incomplete attribute $A_j \in M_i$,

$$\mathsf{can}(t_i) = \{ \prod_{A_j \notin M_i} \{t_i[A_j]\} \times \prod_{A_j \in M_i} \mathsf{can}(t_i[A_j]) \}. \tag{3}$$

Note that both categorical and numerical candidates can be uniformly processed and collaboratively utilized in our work, supported by the normalized distance functions in Section 2.1.1.

Similarly, the *instance candidates* $\mathsf{can}(r)$ of r can be obtained by considering tuple candidates of each tuple $t_i \in r$, i.e., $\mathsf{can}(r) = \prod_{t_i \in r} \mathsf{can}(t_i)$. A filling $r' \in \mathsf{can}(r)$ is also an instance over R, and the *distance cost* in Formula 2 of r' is rewritten as

$$\mathcal{D}(r', C) = \sum_{t'_i \in r'} \Delta(t'_i, C), \tag{4}$$

where $C \subset r'$.

Remark. If we need to cluster data without imputing missing values, our work is still applicable. Imputation candidates bound the possible space of imputing and clustering results. In this sense, our work shows the best result of both imputing and clustering tasks over incomplete data, with the bounded candidates, no matter whether the missing values are imputed at last.

Example 3. *Consider the incomplete instance r in Figure 1(a). Given $k = 5$, the k-nearest neighbors $\{t_6, t_7, t_8, t_9, t_{10}\}$ of t_1 on the complete attribute Pulse suggest cell candidates* $\mathsf{can}(t_1[$*Respiratory Rate*$]) = \{18, 22, 24, 28, 30\}$. *It leads to a number of five tuple candidates, i.e.,* $\mathsf{can}(t_1) = \{(76, 18), (76, 22), (76, 24), (76, 28), (76, 30)\}$. *Similarly, the other incomplete tuples $t_2, t_3, t_4 \in r$ also contain 4, 3, 5 tuple candidates respectively. Considering all the incomplete tuples in r, we have instance candidates $r' \in \mathsf{can}(r)$. For any filling $r' = \{t'_1 = (76, 28), t'_2 = (78, 30), t'_3 = (80, 24), t'_4 = (72, 23), \dots\}$, the distance cost can be calculated according to Formula 4, having $\mathcal{D}(r', C') = 0.29 + 0.30 + 0.43 + \cdots = 5.72$.*

2.2 SCI Problem and Analysis

Considering the possible space of fillings, we study how the imputation and clustering results are affected simultaneously. In this sense, among the instance candidates $r' \in \mathsf{can}(r)$, we find the one with the minimum distance cost $\mathcal{D}(r', C)$.

Problem 1. *Given an incomplete relational instance r with imputation candidates $\mathsf{can}(r)$, the optimal Simultaneous Clustering and Imputing incomplete data (SCI) problem is to find an imputed instance $r' \in \mathsf{can}(r)$ and clustering result C, such that the distance cost $\mathcal{D}(r', C)$ is minimized.*

Unfortunately, we show that the problem is generally hard.

Lemma 1. *The decision version of the optimal SCI problem is NP-complete.*

Remark. Although our study assumes the clustering correlation between missing data and complete data, we argue that it is a widely adopted assumption of existing clustering-based imputation methods with various real-world application scenarios [53, 71, 74]. Meanwhile, we have to admit that our study may not perform very well, when this assumption does not exist in some scenarios, e.g.,

facing serious data heterogeneity and sparsity issues. However, experiments in Section 5.2 over real datasets with various characteristics and real-world missing values show the applicability and generalizability of our work.

Example 4 (Example 3 continued). *Consider another instance candidate* $r'' = \{t'_1 = (76, 24), t'_2 = (78, 24), t'_3 = (80, 24), t'_4 = (86, 23), \dots\}$. *Similarly, according to Formula 4, we can compute the distance cost* $\mathcal{D}(r'', C'') = 0.58 + 0.29 + 0.03 + \cdots = 5.12$, *with* $C'' = \{t'_2, t_{15}\}$. *It is less than that of* $\mathcal{D}(r', C')$ *in Example 3. Actually, given the aforesaid instance candidates* $\mathrm{can}(r)$, r'' *and* C'' *are the optimal results leading to the minimum distance cost.*

3 COMPUTING EXACT SOLUTIONS

In this section, we present an approach for computing the exact solution of the optimal SCI problem, by transforming it into an integer linear programming (ILP) formulation. Given a relational instance r with imputation candidates $\mathrm{can}(r)$ for each tuple $t_i \in r$. Let $c_i = |\mathrm{can}(t_i)|$ denote the number of tuple candidates for $t_i \in r$. Specially, we have $c_i = 1$ for complete tuples $t_i \in r \setminus r_M$, i.e., $\mathrm{can}(t_i) = \{t_i\}$.

Consider the variable $x_{ip} \in \{0, 1\}$, where $x_{ip} = 1$ denotes that the p-th tuple candidate $a_{ip} \in \mathrm{can}(t_i)$ is selected as the filling of t_i, i.e., $t'_i = a_{ip}$; otherwise, $x_{ip} = 0$. The imputation should ensure filling all the missing values. More specifically, for each tuple $t_i \in r$ with candidates $\mathrm{can}(t_i)$, it is required that t_i must be and can only be imputed by one tuple candidate, e.g., $a_{ip} \in \mathrm{can}(t_i)$, having

$$\sum_{p=1}^{c_i} x_{ip} = 1. \tag{5}$$

Let $y_{lq} = 1$ represent that $t'_l = a_{lq}$ acts as a cluster center, i.e., $t'_l (= a_{lq}) \in C$; otherwise, $y_{lq} = 0$. As analyzed in Section 2.1.1, following the same line of partition-based clustering methods [40], we suppose that there are at most κ open cluster centers in C,

$$\sum_{l=1}^{n} \sum_{q=1}^{c_l} y_{lq} \le \kappa. \tag{6}$$

We use the variable z_{iplq} to represent the clustering membership for each imputed tuple $t'_i = a_{ip}$ w.r.t. the corresponding cluster center $t'_l = a_{lq}$. $z_{iplq} = 1$ if $t'_i = a_{ip}$ is assigned to the cluster center $t'_l = a_{lq}$ with $y_{lq} = 1$; otherwise, $z_{iplq} = 0$. Intuitively, one tuple $t'_i \in r'$ can only belong to one cluster, restricting

$$\sum_{p=1}^{c_i} \sum_{l=1}^{n} \sum_{q=1}^{c_l} z_{iplq} = 1. \tag{7}$$

It is notable that the basic preconditions of assigning a tuple $t'_i = a_{ip}$ to the center $t'_l = a_{lq}$, i.e., $z_{iplq} = 1$, are that (i) the tuple candidate a_{ip} is used for imputing t_i ($x_{ip} = 1$); (ii) the candidate a_{lq} is selected as the filling of t_l ($x_{lq} = 1$); and (iii) a cluster center must have been opened at $t'_l = a_{lq}$ ($y_{lq} = 1$).

The first precondition (i) could be represented as the constraint

$$z_{iplq} \le x_{ip}. \tag{8}$$

It ensures that whenever $z_{iplq} = 1$, then we must have $x_{ip} = 1$.

Similarly, the next two preconditions (ii) and (iii) require that $z_{iplq} \le x_{lq}$ and $z_{iplq} \le y_{lq}$. In addition, to make $t'_l = a_{lq}$ become a

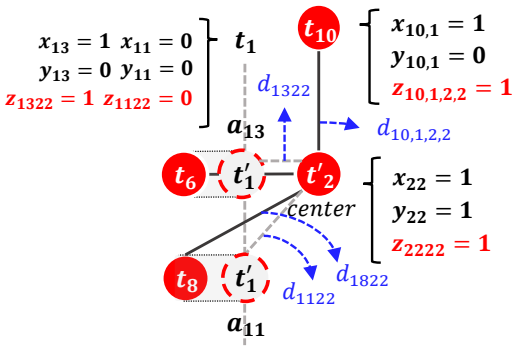

Figure 2: Optimal results of simultaneous clustering and imputing incomplete data by ILP

true cluster center ($y_{lq} = 1$), it also requires t'_l indeed imputed by the tuple candidate a_{lq}, i.e., $x_{lq} = 1$, having $y_{lq} \le x_{lq}$. Combining the aforesaid three constraints, we have

$$z_{iplq} \le y_{lq} \le x_{lq}. \tag{9}$$

Given the constraints in Formulas 5-9, the optimal simultaneous clustering and imputing incomplete data problem can be written as an ILP formulation.

$$\min \sum_{i=1}^{n} \sum_{l=1}^{n} \sum_{p=1}^{c_i} \sum_{q=1}^{c_l} z_{iplq} \mathrm{d}_{iplq} \tag{10}$$

$$\text{s.t.} (5), (6), (7), (8), (9)$$

$$x_{ip}, y_{lq}, z_{iplq} \in \{0, 1\}, \quad 1 \le i \le n, 1 \le p \le c_i, \quad 1 \le l \le n, 1 \le q \le c_l \tag{11}$$

where the weight d_{iplq} for z_{iplq} is a constant denoting the tuple distance between $t'_i = a_{ip}$ and $t'_l = a_{lq}$, i.e., $\mathrm{d}_{iplq} = \Delta(a_{ip}, a_{lq})$.

Remark. Our ILP formulation is different from existing partition-based clustering methods [18, 24, 48, 67], which assume the instance is complete and directly design the ILP formulation over complete tuples. In contrast, we introduce new variables x_{ip} for various imputation candidates $\mathrm{can}(t_i)$, with the new filling constraint (5) and clustering constraints (6)-(9) w.r.t. multiple candidates.

Proposition 2. *The ILP results* $\mathbf{x}^{\mathrm{ILP}}$, $\mathbf{y}^{\mathrm{ILP}}$ *and* $\mathbf{z}^{\mathrm{ILP}}$ *form an optimal clustering solution* C^{ILP} *and imputed instance* r^{ILP} *with the minimum distance cost*

$$\mathcal{D}(r^{\mathrm{ILP}}, C^{\mathrm{ILP}}) = \sum_{i=1}^{n} \sum_{l=1}^{n} \sum_{p=1}^{c_i} \sum_{q=1}^{c_l} z^{\mathrm{ILP}}_{iplq} \mathrm{d}_{iplq},$$

where $t'_i = a_{ip}$ *iff* $x_{ip} = 1$; $y_{lq} \in C$ *iff* $y_{lq} = 1$; *and* $t'_i = a_{ip}$ *belongs to the cluster of* $t'_l = a_{lq}$ *iff* $z_{iplq} = 1$, $1 \le i \le n, 1 \le l \le n, 1 \le p \le c_i, 1 \le q \le c_l$.

Existing ILP solvers [35, 39] can be applied to obtain exact solutions for clustering and imputing results. Unfortunately, it is still intractable to compute the exact solutions. Referring to the previous study [39], the time complexity of computing ILP results is $O(2^{(1-\mathrm{poly}(1/b))n^2c^2})$, where c is the maximum number of tuple candidates for any incomplete tuple in r_M and bn^2c^2 is the number of constraints in ILP.

Example 5. *Consider the relational instance r in Figure 1. We transform the optimal SCI problem into an ILP formulation. For instance, as Figure 2 shows, the tuple candidate $a_{13} \in \mathrm{can}(t_1)$ is associated with a variable x_{13} denoting whether $(76, 24)$ is selected as the filling of t_1, and y_{13} represents whether it will be a cluster center. z_{1322} shows the cluster membership of $t'_1 = a_{13}$ w.r.t. the center t'_2. Moreover, the constant d_{1322} of z_{1322} denotes the tuple distance between $t'_1 = a_{13}$ and $t'_2 = a_{22}$, e.g., $\mathrm{d}_{1222} = 0.43$ and $\mathrm{d}_{1322} = 0.05$. Given $\mathrm{d}_{1322} < \mathrm{d}_{1122}$, by minimizing the objective in Formula 10, a solution of ILP can be $x_{13} = x_{22} = x_{31} = x_{44} = 1$, $y_{15,1} = y_{22} = 1$, $z_{1322} = z_{2222} = z_{3122} = z_{4422} = z_{5122} = \cdots = z_{15,1,15,1} = z_{16,1,15,1} = \cdots = z_{22,1,15,1} = 1$, and the others are 0. It leads to a minimum distance cost, with centers $C = \{t'_2, t_{15}\}$ and clusters $\{\{t'_1, t'_2, t'_3, t'_4, t_5, t_6, \ldots, t_{14}\}, \{t_{15}, \ldots, t_{22}\}\}$.*

4 APPROXIMATION METHODS

Recognizing the hardness of computing exact solutions for the optimal SCI problem, in this section, we focus on designing approximation algorithms. We start by studying heuristics for constructing approximate results based on the linear programming (LP) solution. The idea is to guide the simultaneous clustering and imputing process according to the distance cost W_{ip} for each tuple candidate a_{ip}. Notably, rather than running iteratively as existing studies, the approximation algorithm rounds tuple candidates for only one pass to get clustering and imputing results simultaneously (Proposition 3), with the approximation guarantee (Proposition 4).

Considering the expensive cost to call LP solvers, we further study the efficient initialization for W_{ip} by using local neighbors (LN) without calculating LP solutions. Such a strategy makes the approximation method more efficient (Proposition 5), with the guaranteed performance under the certain case (Proposition 6).

4.1 Heuristics via LP Relaxation

Referring to the difficulty of directly solving the ILP problem to get exact solutions [57], we study the heuristics for constructing approximate solutions that might achieve a near-optimal result efficiently. A natural intuition is to consider an LP relaxation, which is known can be solved in polynomial time [43]. That is, change the requirement of $x_{ip}, y_{lq}, z_{iplq} \in \{0, 1\}$ in Formula 11 to

$$x_{ip}, y_{lq}, z_{iplq} \in [0, 1]. \tag{12}$$

Since the outputs x_{ip}, y_{lq} and z_{iplq} of LP might be fractional, we need to transform the solution into integers to get the clustering result. Inspired by the efficient clustering approach [48], we will show how to round these fractional results of LP into a near-optimal ILP resolution. According to Formula 10, in the LP solution, a filling $t'_i = a_{ip}$ incurs the distance cost

$$W_{ip} = \sum_{l=1}^{n} \sum_{q=1}^{c_l} z_{iplq} \mathrm{d}_{iplq}. \tag{13}$$

This might be spread out over multiple cluster centers $t'_l = a_{lq}$ with $z_{iplq} > 0$. Moreover, according to Formula 7, we have $\sum_{p=1}^{c_i} \sum_{l=1}^{n} \sum_{q=1}^{c_l} z_{iplq} = 1$, which makes it intuitive to consider the value of z_{iplq} as a probability distribution between $t'_i = a_{ip}$ and multiple cluster centers $t'_l = a_{lq}$. Then W_{ip} would be the expectation of the distance cost for $t'_i = a_{ip}$.

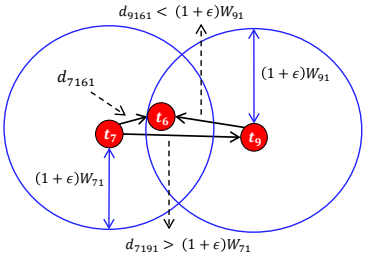

Figure 3: Extended neighborhood for $\epsilon = 0.3$

Tuples with the smallest W_{ip} tend to be similar to the others, which is indeed the intrinsic requirement of cluster centers [40]. Therefore, we heuristically compute the simultaneous clustering and imputing results by estimating the distance cost W_{ip} for each filling candidate a_{ip}. Specifically, we appoint those fillings with the smallest W_{ip} as centers first, which can then be used to cover the other tuples with the imputation closest to them.

Enlightened by this, we define the tuples having distances from $t'_i = a_{ip}$ within W_{ip} as the *neighborhood* $\mathcal{N}(a_{ip}, W_{ip})$ of $t'_i = a_{ip}$,

$$\mathcal{N}(a_{ip}, W_{ip}) = \{a_{lq} \mid t_l \in r, a_{lq} \in \mathrm{can}(t_l), \Delta(a_{ip}, a_{lq}) \leq W_{ip}\}. \tag{14}$$

However, it might be conservative to only consider the neighborhood of $t'_i = a_{ip}$ as its cluster members. As indicated by the well-known clustering method DBSCAN [32], if two points are density-reachable, i.e., existing other points lying in the neighborhoods of both the two points, they will belong to the same cluster. Similarly, we further consider the *extended neighborhood*

$$\mathcal{E}(a_{ip}) = \{a_{lq} \mid t_l \in r, a_{lq} \in \mathrm{can}(t_l), \mathcal{N}(a_{ip}, (1+\epsilon)W_{ip}) \cap \mathcal{N}(a_{lq}, (1+\epsilon)W_{lq}) \neq \emptyset\} \tag{15}$$

of $t'_i = a_{ip}$, where $\epsilon > 0$. For simplicity, combined with the triangle inequality of distance metrics, it can be redefined as

$$\mathcal{E}(a_{ip}) = \{a_{lq} \mid t_l \in r, a_{lq} \in \mathrm{can}(t_l), \mathrm{d}_{iplq} \leq (1+\epsilon)W_{ip} + (1+\epsilon)W_{lq}\}. \tag{16}$$

Given the imputed tuple $t'_i = a_{ip}$ with the smallest W_{ip} as a cluster center, we find its extended neighborhood $\mathcal{E}(a_{ip})$ and assign them to the cluster of the center $t'_i = a_{ip}$.

Remark. Notably, while both our work and [59] utilize the ILP and LP solvers to compute solutions, we argue that such a framework is widely considered to solve the optimization problem [20, 56]. More importantly, given the different objectives and constraints above, their ILP formalizations are thus very different, and the approximation methods are also designed diversely based on different heuristics.

Example 6. *Consider the relational instance r in Figure 1. For a tuple candidate $a_{91} \in \mathrm{can}(t_9)$ of the tuple t_9, according to Formula 13, its distance cost is $W_{91} = 0 * 0.15 + 0 * 0.43 + 0 * 0.30 + \cdots = 0.10$. Referring to Formula 14, the neighborhood of a_{91} is $\mathcal{N}(a_{91}, W_{91}) = \{a_{14}, a_{15}, a_{24}, a_{32}, a_{33}, a_{91}, a_{10,1}, a_{11,1}\}$. Similarly, we have $\mathcal{N}(a_{71}, W_{71}) = \{a_{51}, a_{61}, a_{71}\}$. As shown in Figure 3, since we know $\mathcal{N}(a_{91}, (1+\epsilon)W_{91}) \cap \mathcal{N}(a_{71}, (1+\epsilon)W_{71}) = \{a_{13}, a_{22}, a_{61}\}$, i.e., non-null, according to Formula 15, a_{91} belongs to $\mathcal{E}(a_{71})$, and we have the extended neighborhood $\mathcal{E}(a_{71}) = \{a_{13}, a_{22}, a_{61}, a_{91}\}$.*

4.2 SCI with LP Solution

Based on the heuristics of the distance cost W_{ip} and the extended neighborhood $\mathcal{E}(a_{ip})$ in Section 4.1, Algorithm 1 presents the pseudocode for solving the SCI problem with LP results. Notably, rather than conducting the clustering and imputing processes iteratively until convergence, we round W_{ip} values for only one pass, to obtain both clustering and imputing results efficiently.

Algorithm 1: SCI(r, can(r))

Input: an incomplete relational instance r with imputation candidates can(r)
Output: cluster centers C and corresponding members $\mathbf{V}$

1 $\mathbf{z} \leftarrow$ an LP solution by Formula 12 or LN solution by Formula 17;
2 $C \leftarrow \emptyset, r' \leftarrow \emptyset$;
3 **while** $\arg\min_{a_{lq} \in \text{can}(t_l), t_l \in r} W_{lq} < \mathsf{H}$ **do**
4 $\quad t'_l \leftarrow a_{lq}$;
5 $\quad r' \leftarrow r' \cup \{t'_l\}, C \leftarrow C \cup \{t'_l\}$;
6 $\quad W_{lj} \leftarrow \mathsf{H}$, for each $a_{lj} \in \text{can}(t_l)$;
7 $\quad$ **for** *each* $i \in (\Gamma(\mathcal{E}(a_{lq})) \setminus \Gamma(r'))$ **do**
8 $\quad\quad t'_i \leftarrow \arg\min_{a_{ip} \in \text{can}(t_i) \cap \mathcal{E}(a_{lq})} \mathrm{d}_{iplq}$;
9 $\quad\quad r' \leftarrow r' \cup \{t'_i\}, V_l \leftarrow V_l \cup \{t'_i\}$;
10 $\quad\quad W_{ij} \leftarrow \mathsf{H}$, for each $a_{ij} \in \text{can}(t_i)$;
11 **return** C *and* $\mathbf{V}$

Consider $\mathbf{z}$ as an LP result of variables z_{iplq}, which is first computed by calling LP solvers [35, 43] in Line 1. (By ignoring the LN solution which will be introduced in Section 4.3.) Line 3 greedily picks the smallest W_{lq} in each round, and the corresponding tuple candidate a_{lq} is chosen to impute t_l in Line 4. Then all the other candidates $a_{lj} \in \text{can}(t_l)$ cannot contribute to clustering or imputing processes, and thus can be pruned (by setting W_{lj} to a very large constant H) in Line 6. Let Γ denote the function that returns tuple indexes of the given set, e.g., $\Gamma(a_{12}, a_{23}) = \Gamma(a_{12}, a_{13}, a_{23}) = \{1, 2\}$. Lines 7-10 assign the clustering result for each tuple t_i close to $t'_l (= a_{lq})$. Specifically, for each untouched tuple and involved in the extended neighborhood of the cluster center $t'_l (= a_{lq})$, i.e., $\Gamma(\mathcal{E}(a_{lq})) \setminus \Gamma(r')$ in Line 7, the closest candidate a_{ip} to $t'_l (= a_{lq})$ is selected for imputing t_i in Line 8. Then Line 9 adds the imputed tuple t'_i into the cluster member set V_l of the center t'_l, with all the other tuple candidates $a_{ij} \in \text{can}(t_i)$ pruned in Line 10.

It is easy to see that the correctness of Algorithm 1 is ensured, where (1) none of the filling cells can be imputed again over the current assignments, and (2) each tuple $t'_i \in r'$ must only belong to one cluster, i.e., $\Gamma(C \cup \mathbf{V}) = \Gamma(r)$ and $V_i \cap V_j = \emptyset$ for any $i \neq j$.

Remark. Note that the selection of W_{lq} in Line 3 has no contradiction with the value of x_{lq}. The reason is that the constraint in Formula 8 just ensures if $z_{iplq} = 1$, then a_{ip} must have already been chosen as the imputation for t_i, i.e., $x_{ip} = 1$, instead of a strict equal requirement between W_{lq} and x_{lq}. For instance, even though $z_{iplq} = 0$, x_{ip} can still be 1, since $t'_i = a_{ip}$ can be assigned to any other cluster $t'_j = a_{jk}, 1 \leq j \leq n, j \neq l, 1 \leq k \leq c_j$.

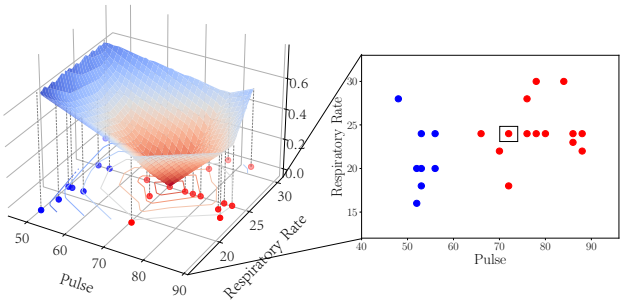

Figure 4: The contour plot of tuple distances to t_6

The time costs of Algorithm 1 with LP solution consist of two parts: (1) computing LP results in Line 1, and (2) constructing clustering and imputing results in Lines 3-10. Given the LP result, it is notable that the clustering process is conducted heuristically for only one pass, instead of designed iteratively as usual studies for incomplete data imputation and clustering [30, 75].

Proposition 3. *The time complexity of Algorithm 1 with LP solution (SCI-LP) is $O(\alpha^{3.5} + (1 + \frac{1}{\epsilon})nm\kappa c)$, where α is the dimension of the LP problem, $n = |r|$, $m = |R|$, κ is the maximum number of clusters, and c is the maximum number of tuple candidates for any incomplete tuple in r.*

In addition to the time complexity, the approximation performance of Algorithm 1 with LP solution is also studied. Let $\mathcal{D}(r', C)$ denote the distance cost of the clustering and imputing results returned by Algorithm 1, and $\mathcal{D}(r^*, C^*)$ be that of the optimal solution r^* and C^* for the SCI Problem 1. We show that the approximation performance is bounded.

Proposition 4. *Algorithm 1 with LP solution (SCI-LP) returns a clustering and imputing result with the distance cost $\mathcal{D}(r', C) \leq 2(1 + \epsilon)\mathcal{D}(r^*, C^*)$.*

Example 7. *Consider the relational instance r in Figure 1. (By ignoring the LN solution which will be introduced later.) Line 1 first computes the LP solution, having $x_{13} = x_{22} = x_{31} = x_{44} = 1$, $x_{11} = x_{12} = \cdots = x_{45} = \cdots = 0$, $z_{1322} = z_{2222} = \cdots = 1$, $z_{1151} = z_{1161} = z_{1171} = \cdots = z_{1122} = \cdots = z_{4522} = \cdots = 0$. Then the imputation a_{22} of t_2 with the minimum $W_{22} = 0$ is selected in Line 3. Since its extended neighborhood is $\mathcal{E}(a_{22}) = \{a_{13}, a_{31}, \dots\}$, the corresponding tuple indexes are $\Gamma(\mathcal{E}(a_{22})) \setminus \Gamma(r') = \{1, 3, \dots\}$. We thus consider the cluster membership for t_2 in Line 7. Among tuple candidates can(t_1), the filling a_{13}, in the extended neighborhood $\mathcal{E}(a_{22})$ with the minimum distance cost d_{1322}, is determined to impute t_1. Finally, it leads to $C = \{t'_2, t_{15}\}$ and clusters $\{\{t'_1, t'_2, t'_3, t'_4, t_5, t_6, \dots, t_{14}\}, \{t_{15}, \dots, t_{22}\}\}$.*

4.3 SCI with LN Solution

Although using the LP relaxation of ILP could make the optimal SCI problem become polynomial-time solvable, it is worth noting that computing the LP solution to initialize the distance cost W_{ip} is still costly (in $O(\alpha^{3.5})$ time). Therefore, in this section, we study performing the more efficient initialization for W_{ip}.

Intuitively, a cluster center would be similar to its members, so that the distance cost in Formula 2 is minimized. For instance, consider the Horse data in Figure 1, where only nearest neighbors like $\{t_9, t_{10}\}$ of the cluster center t'_2 will cause distance costs. On the other hand, those tuples far from t'_2, such as t_{17}, do not contribute

to the cost. Similar results can also be observed in Figure 4, which presents the distances from t_6 to the other points in Figure 1, denoted by the values in z-axis. As shown, only the nearest points to t_6 can cause the cost and form the cluster with it.

Consequently, instead of computing LP solutions and summarizing distances to all the points in Formula 13, we only consider the distances to local neighbors (LN) for initializing W_{ip}. That is, for each tuple candidate $a_{ip} \in \mathsf{can}(t_i)$, $t_i \in r$, we consider its ℓ-nearest-neighbors from $\mathsf{can}(r)$, denoted by $\mathcal{L}(a_{ip})$. Then the LN solution of variables z_{iplq} could be calculated accordingly,

$$z_{iplq} = \begin{cases} \frac{1}{c_i \ell}, & a_{lq} \in \mathcal{L}(a_{ip}) \\ 0, & \text{otherwise} \end{cases}, \tag{17}$$

where $c_i = |\mathsf{can}(t_i)|$. It could be utilized in Line 1 in Algorithm 1 without calling the LP solver, to initialize $\mathbf{z}$ values efficiently. Then the following processes are the same as those analyzed in Section 4.2.

Without calling the costly LP solver, Algorithm 1 becomes more efficient in computing the clustering and imputing results.

Proposition 5. *Algorithm 1 spends $O(n^2c^2(m+\log \ell))$ time computing the LN solution in Line 1, where $n = |r|$, $m = |R|$, ℓ is the number of local neighbors, and c is the maximum number of tuple candidates for any incomplete tuple in r.*

Besides the complexity analysis, we study the approximation performance for Algorithm 1 with LN solution as well. Notably, with the efficient LN solution, Algorithm 1 can still provide an approximation guarantee under the certain case.

Proposition 6. *Algorithm 1 with LN solution (SCI-LN) returns a clustering and imputing result with the distance cost $\mathcal{D}(r', C) \leq 2(1+\epsilon)\mathcal{D}(r^*, C^*)$ for $\ell = 1$.*

Similar to the explanations about Algorithm 1 with LP solution (SCI-LP) in Section 4.2, the correctness of SCI-LN is also ensured. The reason is that (1) each incomplete tuple $t_i \in r_M$ can and only can be imputed by one candidate $a_{ip} \in \mathsf{can}(t_i)$ once, and (2) each imputed tuple $t'_i \in r'$ belongs to one individual cluster exactly.

Example 8. *Rather than computing the LP solution in Example 7, we could directly compute the LN solution in Line 1 in Algorithm 1. For instance, for a tuple candidate $a_{41} \in \mathsf{can}(t_4)$, we can obtain $z_{4,1,12,1} = 0.03, z_{4,1,13,1} = 0.03, z_{4,1,14,1} = 0.03, \ldots$, according to Formula 17. Then the computation procedure in Lines 3-10 is similar to that in Example 7, which returns the final result $C = \{t'_2, t_{15}\}$ and clusters $\{\{t'_1, t'_2, t'_3, t'_4, t_5, t_6, \ldots, t_{14}\}, \{t_{15}, \ldots, t_{22}\}\}$.*

5 EXPERIMENTS

Our experimental objectives are validating two aspects. (1) Could clustering and imputing incomplete data truly improve each other? (2) Are our methods applicable to real applications? To that end, we experimentally investigate the clustering and imputing accuracy, as well as the anomaly detection application performance.

The source code and data are available online [8].

5.1 Experimental Settings

All programs are implemented in Python and experiments are performed on a machine with 2.90GHz CPU and 256GB memory.

5.1.1 Datasets. To evaluate the performance of both clustering and imputing tasks, we utilize the datasets **Banknote** [2], **LED** [16] and **Ecoli** [3] with randomly injected missing values. **Banknote** is constructed based on the image analysis of paper currency and is designed for predicting the authenticity of banknotes. **LED** [16] comprises seven attributes, each representing the display status of a corresponding light-emitting diode segment, with ten clusters denoting the numbers zero to nine. **Ecoli** [3] contains four attributes describing proteins, such as signal sequence recognition, membrane-spanning region prediction, and amino acid content. It can cluster the localization sites of proteins.

To investigate the practicality on serving real scenarios, we use four incomplete datasets with real-world missing values. Although the ground truth is unknown, they contain clustering labels for each tuple. Therefore, we could experimentally study the clustering performance against baselines over them. Specifically, **CRX** [1] contains the credit card information over 15 features, with 5.4% real missing values. **Dermatology** [15] focuses on six kinds of differential diagnosis of erythemato-squamous diseases in dermatology. 2.2% real-world missing data exist in this dataset. **Horse** [14] is about determining the surgical lesion for a horse, helping to figure out whether the sore of a horse needs surgery or not. 98.1% real missing values appear in 23 different attributes. **Soybean** [5] provides insights into the health conditions of soybean plants, with 13.4% real missing values. It encompasses a total of 35 distinct attributes, with the clustering label indicating 19 types of diseases.

In addition, to study the applicable capability of our simultaneous imputing and clustering methods, we also measure the downstream anomaly detection application over **Solar Flare** [4] dataset. It comprises information on solar flares occurring within a 24-hour period, consisting of 1389 instances with 10 features.

5.1.2 Criteria. **Root Mean Square Error (RMSE)** [21] is used to assess the imputation performance, gauging the disparity between imputed values and ground truth.

We employ four widely used metrics with diverse characteristics to evaluate the clustering performance. **Purity** [36] counts the number of tuples from the most common class (truth cluster) for each cluster. **F1** [34] combines the precision and recall, offering a balanced assessment of the clustering quality. **Rand Index (RI)** [68] measures the similarity between true and predicted clusters by considering agreements and disagreements between pairs of tuples. **Adjusted Rand Index (ARI)** [45] gauges the similarity between true and predicted clusters while accounting for chance.

As for the anomaly detection application performance, **Area Under the Curve (AUC)** [22] is adopted as the metric.

5.1.3 Baselines. We consider various competing methods.

Imputation: We select the representative imputation methods, including neighbor-based kNNE [31], statistics-based ERACER [52], multiple imputation MICE [65], clustering-based GMM [71], CMI [74] and IFC [53], ML-based MForest [63] and the widely used ML system HC [55, 69] with constraints, as well as the DL-based GAIN [73] and CSDI [64]. We use the open source implementation in sklearn library [17] for MICE, and the latest release [10] for HC, [13] for MForest, [12] for GAIN, and [7] for CSDI.

Missing Data Tolerant Clustering: To investigate the effect of directly clustering without filling missing values, we also compare

Table 1: Clustering performance on raw data without/with imputation over various real-world incomplete datasets

Methods	CRX					Dermatology					Horse					Soybean				
	Pur.	*F1*	*RI*	*ARI*	*Time*	*Pur.*	*F1*	*RI*	*ARI*	*Time*	*Pur.*	*F1*	*RI*	*ARI*	*Time*	*Pur.*	*F1*	*RI*	*ARI*	*Time*
Raw	0.728	0.494	0.565	0.132	0.992	0.866	0.775	0.910	0.719	0.133	0.706	0.597	0.584	0.167	0.278	0.676	0.424	0.912	0.377	1.439
kNNE	0.727	0.491	0.565	0.132	1.092	0.866	0.776	0.911	0.720	0.224	0.767	0.655	0.642	0.283	0.314	0.689	0.436	0.911	0.388	3.203
ERACER	0.764	0.533	0.595	0.192	43.30	0.866	0.780	0.912	0.725	3.992	0.704	0.596	0.582	0.165	0.364	0.674	0.433	0.913	0.387	106.6
MICE	0.734	0.501	0.576	0.154	0.949	0.866	0.776	0.911	0.720	0.461	0.731	0.618	0.606	0.211	0.514	0.679	0.432	0.913	0.386	2.002
GMM	0.748	0.513	0.577	0.156	4.205	0.866	0.781	0.912	0.726	3.594	0.746	0.637	0.620	0.238	2.240	0.692	0.435	0.914	0.389	18.19
CMI	0.763	0.528	0.596	0.195	1.508	0.866	0.779	0.912	0.724	1.578	0.745	0.637	0.619	0.237	0.141	0.669	0.424	0.910	0.375	4.136
IFC	0.772	0.539	0.606	0.214	1.658	0.866	0.780	0.912	0.725	1.081	0.774	0.663	0.650	0.299	0.439	0.691	0.437	0.910	0.388	2.047
MForest	0.719	0.484	0.559	0.122	8.977	0.825	0.696	0.881	0.622	1.277	0.641	0.556	0.539	0.077	44.51	0.684	0.417	0.908	0.367	56.99
HC	0.775	0.541	0.603	0.208	28.40	0.858	0.752	0.901	0.690	15.03	0.666	0.569	0.554	0.107	43.17	0.681	0.430	0.909	0.381	64.51
GAIN	0.719	0.484	0.559	0.121	174.2	0.861	0.757	0.903	0.696	98.83	0.633	0.559	0.534	0.066	108.1	0.681	0.429	0.912	0.382	115.5
CSDI	0.730	0.373	0.552	0.110	183.9	0.809	0.689	0.876	0.612	168.0	0.764	0.396	0.549	0.131	128.8	0.645	0.434	0.912	0.387	156.6
NMF	0.693	0.527	0.546	0.093	50.19	0.557	0.540	0.707	0.367	3.789	0.690	0.590	0.571	0.141	1.393	0.469	0.345	0.854	0.269	0.283
kPOD	0.766	0.548	0.606	0.214	0.171	0.767	0.647	0.849	0.552	0.082	0.652	0.627	0.536	0.043	0.097	0.646	0.437	0.898	0.381	0.570
CI	0.768	0.526	0.595	0.192	13.83	0.866	0.778	0.911	0.722	2.307	0.734	0.622	0.609	0.218	1.659	0.678	0.436	0.913	0.389	35.66
DPCI	0.593	0.538	0.017	0.016	3.986	0.503	0.451	0.516	0.207	1.194	n/a	n/a	n/a	n/a	n/a	0.577	0.415	0.831	0.336	2.790
kCMM	0.690	0.577	0.571	0.143	1192	0.316	0.228	0.654	0.009	70.98	0.630	0.517	0.499	0.001	913.3	0.529	0.317	0.878	0.251	2008
GMMC	0.628	0.409	0.546	0.090	0.264	0.762	0.687	0.860	0.599	0.119	0.631	0.493	0.508	-0.02	11.82	0.659	0.444	0.908	0.395	1.045
SCI-ILP	0.794	0.674	0.673	0.345	32.684	0.926	0.886	0.955	0.858	14.113	0.785	0.675	0.662	0.323	16.104	0.788	0.479	0.927	0.442	11.248
SCI-LP	0.794	0.674	0.673	0.345	23.07	0.907	0.853	0.943	0.818	6.307	0.783	0.672	0.659	0.317	11.01	0.765	0.471	0.925	0.432	8.117
SCI-LN	0.780	0.586	0.631	0.265	6.038	0.847	0.787	0.913	0.732	1.627	0.780	0.672	0.656	0.304	2.494	0.704	0.437	0.923	0.397	3.556

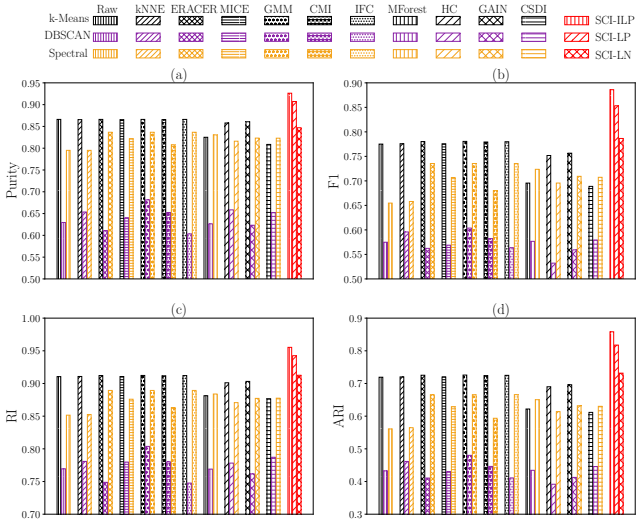

Figure 5: Clustering performance on raw data without/with various imputation methods, complementary with different clustering approaches, over Dermatology

with missing data tolerant clustering methods NMF [44] and kPOD [27], with the open source implementation [11] for kPOD.

Unified Framework: The most relevant studies CI [54], DPCI [33], kCMM [30], and GMMC [75] integrate imputation and clustering into one framework. We use the open source implementation [9] for GMMC and [6] for kCMM.

5.2 Clustering Comparison with Baselines

Although missing data tolerant approaches and unified frameworks could directly provide clustering results over incomplete data, imputation methods can only impute missing values and the clustering results are obtained with the help of the other clustering methods. Therefore, we first study the clustering performance for imputation methods with varying clustering approaches in Figure 5. We use various imputation baselines to fill missing values first, then conduct the frequently used k-Means [40], DBSCAN [32], and Spectral Clustering [28] approaches over the imputed data respectively. We can observe that k-Means clustering generally achieves the best performance for diverse imputation methods, followed by Spectral Clustering, while DBSCAN performs relatively worse compared to the other two. The reason is that the density-based clustering method DBSCAN may result in unclear cluster boundaries or failure to correctly cluster all points on datasets with uneven densities. Moreover, in the high-dimensional space, e.g., the Dermatology dataset comprising 34 symptom features to determine the types of Eryhemato-Squamous Disease, distances between data points can become very close, making it difficult to distinguish similarity values in the similarity matrix. Meanwhile, in addition to DBSCAN, this sparsity of data and distance distortion can also affect the performance of Spectral Clustering, leading to inaccurate clustering results. In contrast, k-Means can be adapted to different applications and used with different distance metrics to cluster sparse high-dimensional data. Beyond that, DBSCAN is sensitive to parameters, and determining appropriate values for the density-related parameters, e.g., the distance threshold *eps* and density threshold *minPts*, can be challenging. Spectral Clustering, on the other hand, lacks computational efficiency due to its higher complexity. Conversely, k-Means is simple, efficient, and has low computational costs, making it widely used in practical applications. Similar results to Figure 5 also appear over the other datasets in experiments, which verify our intuition that the objective function is designed following partition-based clustering methods in Definition 1 as well. Therefore, we use the k-Means clustering for imputation baselines in the following experiments by default.

Table 1 presents clustering results on raw incomplete data without/with competitive imputation methods, missing data tolerant clustering approaches, unified frameworks, and our SCI algorithms, over real-world incomplete datasets. As shown, the clustering performance after data imputation is generally better than directly

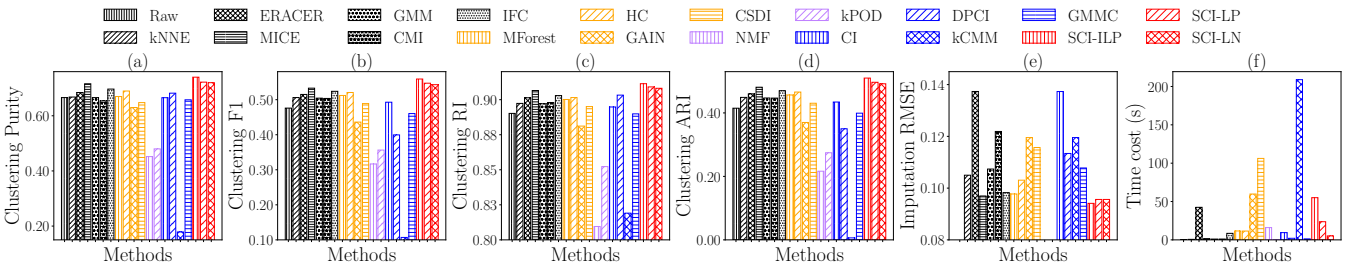

Figure 6: Clustering and imputation performance on raw data without/with imputation over LED

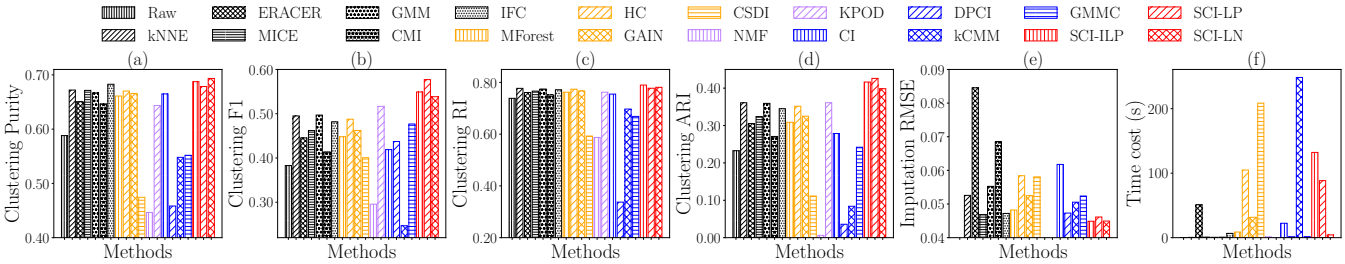

Figure 7: Clustering and imputation performance on raw data without/with imputation over Ecoli

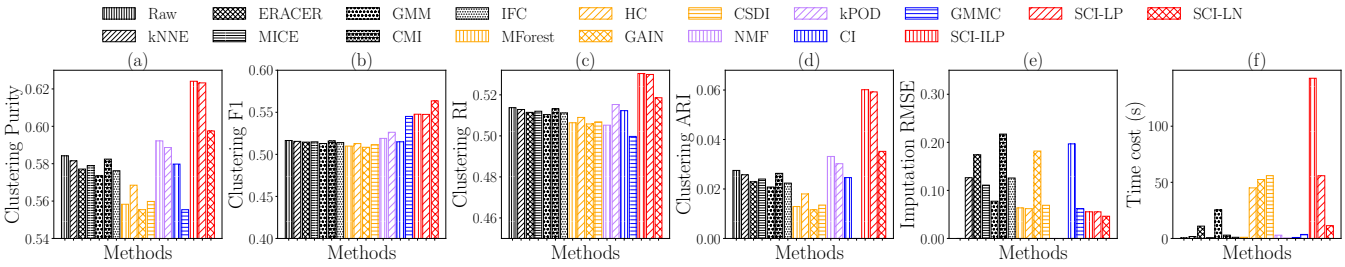

Figure 8: Clustering and imputation performance on raw data without/with imputation over Banknote

performing over raw data. Such results show the necessity of imputing missing data for better clustering.

The general-purpose data imputation methods [31, 52, 53, 55, 63–65, 69, 71, 73, 74] do not consider the improvement of the downstream clustering performance. For instance, as shown in Figure 1, while the imputation methods fill missing values to meet the cleaning criteria, the imputed data may not always bring a positive impact on the clustering result. Actually, the imputation methods may fill all the missing data to optimize various objectives, which are unrelated to the clustering performance. It explains why existing imputation methods cannot consistently perform well in improving the clustering performance after imputing missing values.

It is not surprising that missing data tolerant clustering methods NMF and kPOD perform worse than existing imputation baselines in most cases. The reason is that their clustering operations are directly conducted on the incomplete raw data, with too much information loss caused by missing values. Such results verify our intuition again that imputing missing values is very important for improving the clustering performance.

Although the unified frameworks CI, DPCI, kCMM and GMMC integrate clustering and imputing incomplete data tasks into one framework and process them iteratively, we find that the imputed values may not be able to necessarily improve the clustering performance and may even hinder it, because of the unclear correlation between the fillings and clustering results. Besides, the performance of GMMC is greatly associated with that of GMM. If the true data distribution does not match the Gaussian distribution, the imputed values may be biased. Further, clustering results based on such biased fillings and iterating alternately may amplify the biases, which could be a reason why the performance of GMMC is sometimes inferior to that of GMM. Specially, since DPCI constructs clustering and imputation models based on complete tuples, we omit its experimental result over Horse dataset, where all tuples contain more or less missing values.

In contrast, our SCI conducts simultaneous clustering and imputing over incomplete data, generally achieving the best overall clustering performance in most cases. The results demonstrate the contribution of simultaneous clustering and imputing incomplete data, to take full advantage of their mutual benefits.

As for time costs, the ML system HC, DL models GAIN and CSDI, as well as iterative approaches such as ERACER and kCMM, are slower than the others. SCI-LN is more efficient than most existing studies and our SCI-LP and SCI-ILP, which is not surprising referring to the non-iterative design with the complexity analysis in Proposition 3. Nevertheless, the clustering performance of SCI-LP

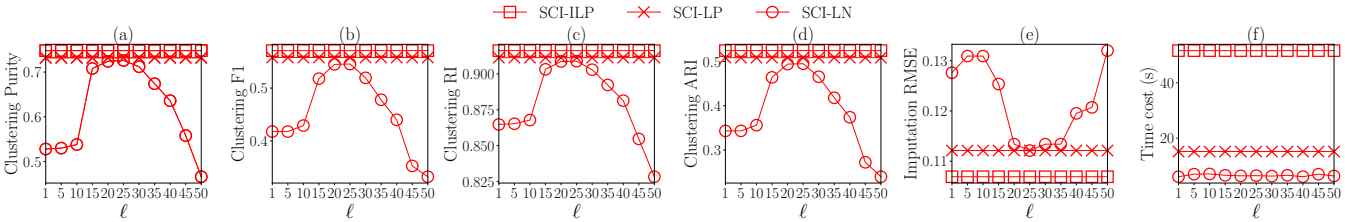

Figure 9: Clustering and imputation performance with varying the number of local neighbors ℓ over LED

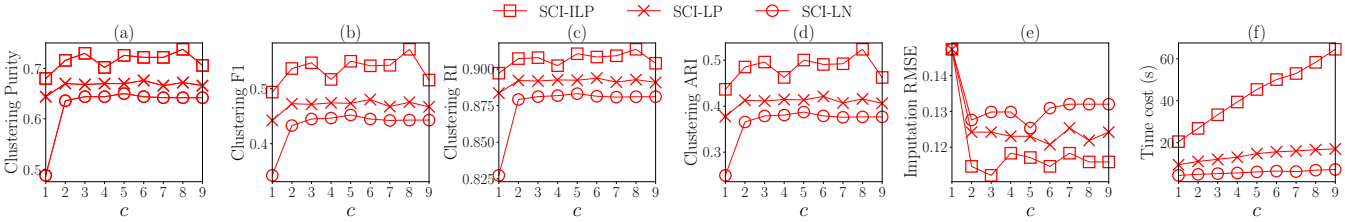

Figure 10: Clustering and imputation performance with varying the number of candidates c over LED

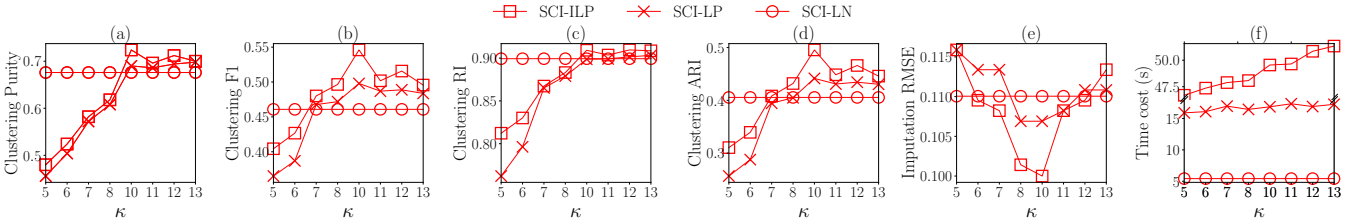

Figure 11: Clustering and imputation performance with varying the number of considered clusters κ over LED

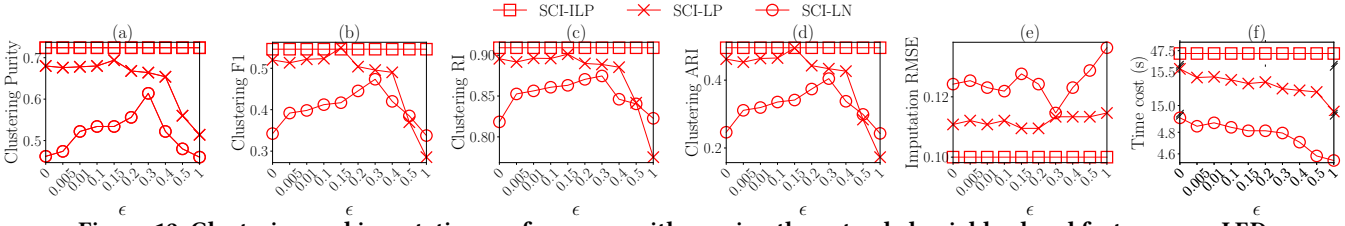

Figure 12: Clustering and imputation performance with varying the extended neighborhood factor ϵ over LED

and SCI-ILP is better than SCI-LN in most tests in Table 1. Therefore, we may use SCI-LP and SCI-ILP in small datasets for high accuracy, and SCI-LN is preferable for large datasets as a good trade-off between effectiveness and efficiency.

5.3 Clustering and Imputing Comparison

To better investigate the mutual benefits of clustering and imputation tasks, we experimentally evaluate both clustering and imputing performance over LED, Ecoli and Banknote datasets in Figures 6-8. As shown in Figures 6(e), 7(e), 8(e), imputation methods based on machine learning or deep learning (with yellow legends) can often generate better fillings than traditional approaches (with black legends), due to their ability to explore more complex patterns and dependencies. However, this comes at higher time costs in Figures 6(f), 7(f), 8(f). On the other hand, as for the clustering performance, machine learning or deep learning based imputation methods do not exhibit more obvious superiority than traditional approaches, e.g., as shown in Figures 8(a)-(d). Such results show that existing imputation techniques do not always improve the clustering performance and may even have a negative impact.

Since missing data tolerant clustering methods NMF and kPOD cannot impute missing values, we only compare their clustering performance in subfigures (a)-(d), and do not assess the imputation RMSE in the subfigure (e). Additionally, owing to the unbalanced cluster issue present in DPCI and kCMM methods on the Banknote dataset (i.e., categorizing instances entirely or overwhelmingly into one cluster), these two methods are omitted in Figure 8. Considering that NMF and kPOD directly clustering incomplete data may suffer from much information loss, it is not surprising that they cannot achieve a good clustering result. Meanwhile, although CI, DPCI, kCMM, and GMMC integrate clustering and imputation tasks into a unified framework, they do not establish a clear connection between these two tasks. Consequently, the performance of imputation and clustering tasks is not consistent. For example, although GMMC may perform well when imputing missing values for the Banknote dataset in Figure 8(e), its corresponding clustering result is relatively inaccurate in Figures 8(a)-(d).

On the contrary, our algorithms conduct clustering and imputing processes simultaneously, to take full advantage of their mutual

benefits. Theoretically, more accurate imputation results can provide more valuable information to guide the clustering process, and more accurate clustering results can offer more reliable evidence to impute missing data in return. When clustering tuples into different clusters, the tuple values within the same cluster are often similar, thus enabling mutual reference for imputing incomplete tuples. Therefore, rather than treating them independently or conducting them respectively, we combine clustering and imputing targets into one single objective collaboratively, to achieve the optimal results for both tasks. The clustering and imputing results in Figures 6-8 verify the significance of such designs, where our methods show a generally better performance on both tasks.

In Figure 8, we study both clustering and imputation performance over the Banknote dataset with 20% missing values. Owing to manufacturing errors, print quality, anti-counterfeiting techniques, etc, Banknote dataset may contain heterogeneous values among tuples. We thus consider more neighbors of the incomplete tuple to provide candidates. Then the time cost of SCI-ILP may exceed that of other methods. Actually, if we use more advanced approaches (instead of simply computing similar neighbors) to provide candidates, the candidate set can be significantly narrowed down. Such results also prompt us to design more efficient approximate algorithms SCI-LP and SCI-LN. As shown, our SCI-LN is more efficient than most baselines and our SCI-LP and SCI-ILP, with good results in clustering and imputing tasks.

5.4 Performance of Our Algorithms

Figure 9 presents the performance sensitivity results with various numbers of local neighbors ℓ. Since only SCI-LN requires such a parameter, SCI-ILP and SCI-LP are insensitive to ℓ in Figure 9. As shown in Figures 9(a)-(e), the clustering and imputation performance of SCI-LN generally exhibits a trend from increasing to decreasing as ℓ enlarges. This occurs because SCI-LN considers a few local neighbors when ℓ is small, resulting in a tuple candidate $a_{ip} \in \mathsf{can}(t_i)$ containing only those neighbors very close to it. Such a small number of neighbors are not enough to represent all the characteristics of the cluster to which the tuple candidate a_{ip} belongs. Conversely, an excessively large ℓ value may lead to overly broad local neighbors, potentially including those tuples that are distant or significantly different from the given tuple candidate a_{ip}. These tuples may come from different clusters with a_{ip} and can contain heterogeneous values between each other.

Figure 10 shows the results of our algorithms with various numbers of candidates c. It is not surprising that the time cost increases with more candidates used, since the larger search space is correspondingly involved in clustering and imputing tasks. Meanwhile, as the number of candidates increases, we observe that SCI-ILP, SCI-LP, and SCI-LN all exhibit an initial improvement in both clustering and imputing performance, followed by a gradual stabilization when the candidate number is sufficiently large. This phenomenon arises because the distribution of a few candidates from different neighbors may be biased, hindering our methods to compute appropriate results. As the number of candidates increases, our methods can explore a wider range of possibilities during imputation and clustering processes, resulting in more accurate fillings and clusters. Moreover, when the number of candidates is enough, the candidate

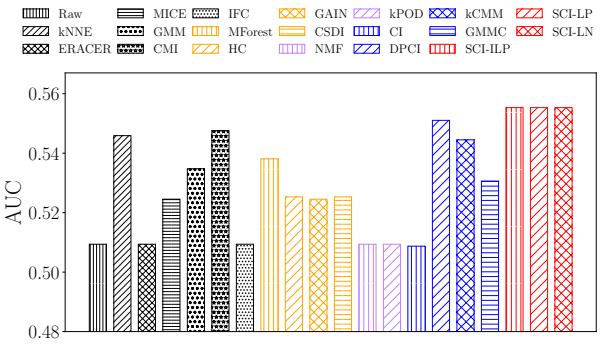

Figure 13: Anomaly detection application performance on raw data without/with imputation over Solar Flare

set already encompasses filling values close to the incomplete tuple. Further expanding the candidate set can only introduce values from tuples that are farther away, thus failing to yield better imputing and clustering results. Consequently, the imputing and clustering performance tends to plateau.

Figure 11 illustrates the clustering and imputation outcomes by our algorithms across different numbers of clusters κ. It is not surprising that a more similar number of clusters to the exact situation in the dataset leads to better clustering and imputing performance, as the parameter value of κ increases in Figures 11(a)-(e). The reason is that a small κ value may lead to insufficient clusters for consideration, which affects both clustering and imputation results. Consequently, the performance of SCI-ILP and SCI-LP shows an increasing trend, reaching their peaks near the actual number of clusters in the dataset. However, with the further increase of κ, algorithms may split clusters into finer granularity, deviating from the actual clustering structure in the dataset, thus affecting the experimental performance. Therefore, the clustering and imputation results are not good for extreme cases. Since SCI-LN does not require such a parameter, it is insensitive to κ in Figure 11.

Figure 12 depicts the performance sensitivity results of our algorithms across different extended neighborhood factors ϵ. Since SCI-ILP computes exact solutions without the parameter ϵ, it performs stably in Figure 12. As for SCI-LP and SCI-LN, similarly, we can observe that neither a large nor a small ϵ could lead to the best performance for them in Figures 12(a)-(e). If ϵ is too large, it will mistakenly include tuples from different clusters into the same large cluster. Meanwhile, when ϵ is too small, it severs the connections between clusters, making inaccurate results as well. In particular, when $\epsilon = 0$, only the strict neighborhood in Formula 14 is used to form clusters. The results demonstrate the intuition of considering the extended neighborhood in Formula 15 when constructing clusters. Observing Figures 12(a)-(d) and Figure 12(e), we can note that the performance trends of clustering and imputation tasks are generally consistent. SCI-LP and SCI-LN reach their optimum when ϵ equals 0.15 and 0.3 respectively, which also confirms the significant correlation between clustering and imputation tasks.

5.5 Application Study

To validate whether our methods are practical for real applications, we study anomaly detection application performance over Solar

Flare dataset in Figure 13. We randomly inject 20% missing values and the Isolation Forest algorithm [50] is employed for the anomaly detection. As shown in Figure 13, although the unified frameworks CI, DPCI, kCMM and GMMC also consider both tasks, applying them to practical applications may be hard to yield good results. Considering the comparison in previous clustering and imputing experiments in Table 1 and Figures 6-8, such results on subsequent applications are not surprising. On the contrary, our methods considering simultaneous clustering and imputing over incomplete data are preferable for serving real applications.

6 RELATED WORK

Since this work studies clustering and imputing incomplete data simultaneously, we discuss below the related works in both topics.

6.1 Clustering

Data clustering has been shown to be useful in various applications [41]. Partition-based clustering methods [66] divide the instance into multiple clusters based on their characteristics and similarities, which are most commonly used in real applications. k-Means clustering [18] calculates the mean value for each cluster to determine its center point. As a variation of the k-Means clustering, k-Median clustering [24, 48, 67] calculates the median of one cluster as the cluster center. Although showing promising performance over complete data, these approaches face intractable problems of clustering incomplete data, since the distances between missing values and complete values cannot be effectively measured.

As mentioned above, missing values could prevent the clustering application. Therefore, missing data tolerant clustering methods [27, 44] are designed for clustering incomplete tuples only with their complete values. Although the recent study [62] further considers the subspace clustering, it still only uses complete values and neglects the imputation for incomplete cells. As shown in Section 5.2, with too much information loss, their clustering results are inaccurate. In contrast, our work considers simultaneous clustering and imputing processes, where missing values can be imputed and utilized as a strong support in clustering.

Despite the different data quality issues addressed, i.e., missing values in this study and erroneous data in [58, 59], this manuscript is also different from [58, 59] in criteria, problems and technical aspects. (1) DORC [59] and DISC [58] follow the minimum repair principle, but our SCI utilizes the distance cost as the criteria. (2) DORC requires repaired data to be anomaly-free, while DISC mandates that the repair results adhere to distance constraints. Unlike them, SCI finds imputing and clustering results simultaneously to minimize the distance cost. (3) The devised technical methods are thus very different given distinct criteria and problems. Notably, while both SCI and DORC utilize the ILP and LP solvers, given the different objectives and constraints above, their ILP formalizations and the approximation methods are thus designed diversely.

6.2 Imputing

Existing imputation methods often fill missing data based on various signals, e.g.,constraints, neighbors, clusters, statistics, ML, DL, etc. Constraint-based methods [55, 61] usually employ those values satisfying the constraints to impute missing data. The methods based on statistics [52, 60, 70] capture probabilistic correlations between reliable attributes with complete values and flexible attributes with incomplete values. The imputation is thus to find values that can maximize the likelihood w.r.t. the probabilistic correlations. Neighbor-based approaches [19, 31] utilize the neighbors of each incomplete tuple t_i to impute missing data, where neighbors are determined according to the complete attributes [19] or various subsets of them [31]. Clustering-based methods determine neighbors using clustering results over incomplete tuples, including the kernel function strategy [74], fuzzy k-Means [46] and its iterative manner [53], as well as gaussian mixture models (GMM) [71]. Machine learning and deep learning techniques are used for data imputation recently [23, 26, 64, 73]. Among them, Generative Adversarial Networks (GAN) [64, 73] are widely considered, with the capability of generating filling values. However, without considering the influence on the clustering performance, they cannot necessarily bring a positive impact on the clustering result.

Recent works realize that imputation and clustering tasks of incomplete data should be considered in a unified framework [30, 33, 54, 75], to take full advantage of their mutual benefits. In general, missing data are imputed by clustering results, and the imputed data are then taken for clustering. As analyzed in the Introduction, the major problem of such methods is that there is still no clear connection between imputing and clustering tasks, which are conducted respectively. Therefore, both clustering and imputing processes cannot guarantee to improve each other.

7 CONCLUSIONS

In this paper, inspired by the aforesaid victory and defeat, we consider clustering and imputing incomplete data simultaneously. To ensure the successful improvement between the two processes, we (1) formalize the problem and analyze its NP-hardness in Lemma 1; (2) devise an exact algorithm by transforming the problem into the ILP formulation; (3) design approximation algorithms based on the LP relaxation and the LN solution with approximation performance guarantees in Propositions 4 and 6. Extensive experiments over various datasets with real-world missing values show the superiority of our work in both clustering and imputing incomplete data. The sensitivity experiments of our approaches w.r.t. the key parameters are studied. Moreover, the anomaly detection application study demonstrates the practicality of our methods. For future studies, it is also interesting to consider the other data analytics tasks, e.g., classification and regression, over incomplete data.

ACKNOWLEDGEMENTS

We thank the reviewers for their constructive and professional comments. This work is supported in part by the National Natural Science Foundation of China (62302241, 62372252, 72342017, 62232005, 62072265, 62021002, 92267203), the National Key Research and Development Plan (2021YFB3300500), Beijing Key Laboratory of Industrial Big Data System and Application, Natural Science Foundation of Tianjin (22JCQNJC01520), the Fundamental Research Funds for the Central Universities, Nankai University (63231147). Shaoxu Song is the corresponding author (https://sxsong.github.io/).

REFERENCES

[1] 2024. https://sci2s.ugr.es/keel/dataset.php?cod=59.
[2] 2024. https://archive.ics.uci.edu/dataset/267/banknote+authentication.
[3] 2024. https://archive.ics.uci.edu/dataset/39/ecoli.
[4] 2024. https://archive.ics.uci.edu/dataset/89/solar+flare.
[5] 2024. https://archive.ics.uci.edu/dataset/90/soybean+large.
[6] 2024. https://github.com/ClarkDinh/k-CMM.
[7] 2024. https://github.com/ermongroup/CSDI.
[8] 2024. https://github.com/erssss/impute-SCI.
[9] 2024. https://github.com/ethan-yizhang/GMM-with-Incomplete-Data.
[10] 2024. https://github.com/HoloClean/holoclean.
[11] 2024. https://github.com/iiradia/kPOD.
[12] 2024. https://github.com/jsyoon0823/GAIN.
[13] 2024. https://hackage.haskell.org/package/MissingPy.
[14] 2024. https://sci2s.ugr.es/keel/dataset.php?cod=180.
[15] 2024. https://sci2s.ugr.es/keel/dataset.php?cod=60.
[16] 2024. https://sci2s.ugr.es/keel/dataset.php?cod=63.
[17] 2024. https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html.
[18] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. 2017. Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms. In *FOCS*. 61–72.
[19] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.
[20] Arvind Arasu, Raghav Kaushik, and Jian Li. 2011. Data generation using declarative constraints. In *SIGMOD*. 685–696.
[21] J Scott Armstrong and Fred Collopy. 1992. Error measures for generalizing about forecasting methods: Empirical comparisons. *IJF* 8, 1 (1992), 69–80.
[22] Andrew P Bradley. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *PR* 30, 7 (1997), 1145–1159.
[23] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. 2018. BRITS: Bidirectional Recurrent Imputation for Time Series. In *NeurIPS*. 6776–6786.
[24] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. 2002. A Constant-Factor Approximation Algorithm for the k-Median Problem. *JCSS* 65, 1 (2002), 129–149.
[25] Sanjay Chawla and Aristides Gionis. 2013. k-means−: A unified approach to clustering and outlier detection. In *SIAM*. 189–197.
[26] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *SR* 8, 1 (2018), 1–12.
[27] Jocelyn T Chi, Eric C Chi, and Richard G Baraniuk. 2016. k-pod: A method for k-means clustering of missing data. *The American Statistician* 70, 1 (2016), 91–99.
[28] Anil Damle, Victor Minden, and Lexing Ying. 2019. Simple, direct and efficient multi-way spectral clustering. *Information and Inference* 8, 1 (2019), 181–203.
[29] Marcílio Carlos Pereira de Souto, Pablo A. Jaskowiak, and Ivan G. Costa. 2015. Impact of missing data imputation methods on gene expression clustering and classification. *BMC Bioinform.* 16 (2015), 64:1–64:9.
[30] Duy-Tai Dinh, Van-Nam Huynh, and Songsak Sriboonchitta. 2021. Clustering mixed numerical and categorical data with missing values. *IS* 571 (2021), 418–442.
[31] Carlotta Domeniconi and Bojun Yan. 2004. Nearest Neighbor Ensemble. In *ICPR*. 228–231.
[32] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*. 226–231.
[33] Kun Gao, Hassan Ali Khan, and Wenwen Qu. 2022. Clustering with Missing Features: A Density-Based Approach. *Symmetry* 14, 1 (2022), 60.
[34] Cyril Goutte and Eric Gaussier. 2005. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *ECIR*. Springer, 345–359.
[35] Gurobi Optimization, LLC. 2021. Gurobi Optimizer Reference Manual.
[36] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann.
[37] Ihab F. Ilyas and Xu Chu. 2019. *Data Cleaning*. ACM.
[38] Ihab F. Ilyas and Theodoros Rekatsinas. 2022. Machine Learning and Data Cleaning: Which Serves the Other? *JDIQ* 14, 3 (2022), 13:1–13:11.
[39] Russell Impagliazzo, Shachar Lovett, Ramamohan Paturi, and Stefan Schneider. 2014. 0-1 Integer Linear Programming with a Linear Number of Constraints. *CoRR* abs/1401.5512 (2014). arXiv:1401.5512
[40] Anil K. Jain and Richard C. Dubes. 1988. *Algorithms for Clustering Data*.
[41] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. 1999. Data Clustering: A Review. *ACM Comput. Surv.* 31, 3 (1999), 264–323.
[42] Bojan Karlas, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. 2020. Nearest Neighbor Classifiers over Incomplete Information: From Certain Answers to Certain Predictions. *PVLDB* 14, 3 (2020), 255–267.
[43] Narendra Karmarkar. 1984. A New Polynomial-Time Algorithm for Linear Programming. In *STOC*. 302–311.
[44] Jingu Kim and Haesun Park. 2008. *Sparse nonnegative matrix factorization for clustering*. Technical Report. Georgia Institute of Technology.
[45] Hubert Lawrence and Arabie Phipps. 1985. Comparing partitions. *Journal of classification* 2, 1 (1985), 193–218.
[46] Dan Li, Jitender S. Deogun, William Spaulding, and Bill Shuart. 2004. Towards Missing Data Imputation: A Study of Fuzzy K-means Clustering Method. In *RSCTC*, Vol. 3066. 573–579.
[47] Yujian Li and Bi Liu. 2007. A Normalized Levenshtein Distance Metric. *TPAMI* 29, 6 (2007), 1091–1095.
[48] Jyh-Han Lin and Jeffrey Scott Vitter. 1992. Approximation Algorithms for Geometric Median Problems. *Inf. Process. Lett.* 44, 5 (1992), 245–249.
[49] Roderick JA Little and Donald B Rubin. 2019. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.
[50] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *ICDM*. 413–422.
[51] John David MacCuish and Norah E MacCuish. 2010. *Clustering in bioinformatics and drug discovery*. CRC Press.
[52] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: a database approach for statistical inference and data cleaning. In *SIGMOD*. 75–86.
[53] Sanaz Nikfalazar, Chung-Hsing Yeh, Susan E. Bedingfield, and Hadi Akbarzade Khorshidi. 2017. A new iterative fuzzy clustering algorithm for multiple imputation of missing data. In *FUZZ-IEEE*. 1–6.
[54] OLIVER Pfaffel. 2020. ClustImpute: An R package for K-means clustering with build-in missing data imputation.
[55] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201.
[56] Theodoros Rekatsinas, Amol Deshpande, and Ashwin Machanavajjhala. 2013. SPARSI: Partitioning sensitive data amongst multiple adversaries. *PVLDB* 6, 13 (2013), 1594–1605.
[57] Hossein Shahrabi Farahani and Jens Lagergren. 2013. Learning oncogenetic networks by reducing to mixed integer linear programming. *PloS one* 8, 6 (2013), e65773.
[58] Shaoxu Song, Fei Gao, Ruihong Huang, and Yihan Wang. 2021. On Saving Outliers for Better Clustering over Noisy Data. In *SIGMOD*. 1692–1704.
[59] Shaoxu Song, Chunping Li, and Xiaoquan Zhang. 2015. Turn Waste into Wealth: On Simultaneous Clustering and Cleaning over Dirty Data. In *SIGKDD*. 1115–1124.
[60] Shaoxu Song and Yu Sun. 2020. Imputing Various Incomplete Attributes via Distance Likelihood Maximization. In *SIGKDD*. 535–545.
[61] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2020. Enriching Data Imputation under Similarity Rule Constraints. *TKDE* 32, 2 (2020), 275–287.
[62] Akhilesh Soni, Jeff Linderoth, Jim Luedtke, and Daniel Pimentel-Alarcón. 2023. An Integer Programming Approach To Subspace Clustering With Missing Data. *arXiv preprint arXiv:2309.15285* (2023).
[63] Daniel J. Stekhoven and Peter Bühlmann. 2012. MissForest - non-parametric missing value imputation for mixed-type data. *Bioinform.* 28, 1 (2012), 112–118.
[64] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *NeurIPS*. 24804–24816.
[65] Stef Van Buuren and Karin Groothuis-Oudshoorn. 2011. mice: Multivariate imputation by chained equations in R. *JOSS* 45 (2011), 1–67.
[66] T Velmurugan and T Santhanam. 2011. A survey of partition based clustering algorithms in data mining: An experimental approach. *Information Technology Journal* 10, 3 (2011), 478–484.
[67] Christopher Whelan, Greg Harrell, and Jin Wang. 2015. Understanding the k-medians problem. In *CSC*. 219.
[68] M William. 1971. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.
[69] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *MLSys*.
[70] Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *SIGMOD*. 553–564.
[71] Xiaobo Yan, Weiqing Xiong, Liang Hu, Feng Wang, and Kuo Zhao. 2015. Missing value imputation based on gaussian mixture model for the internet of things. *MPE* (2015).
[72] Yihang Yin, Fengzheng Liu, Xiang Zhou, and Quanzhong Li. 2015. An efficient data compression model based on spatial clustering and principal component analysis in wireless sensor networks. *Sensors* 15, 8 (2015), 19443–19465.
[73] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *ICML*. 5675–5684.
[74] Shichao Zhang, Jilian Zhang, Xiaofeng Zhu, Yongsong Qin, and Chengqi Zhang. 2008. Missing Value Imputation Based on Data Clustering. *TCS* 1 (2008), 128–138.
[75] Yi Zhang, Miaomiao Li, Siwei Wang, Sisi Dai, Lei Luo, En Zhu, Huiying Xu, Xinzhong Zhu, Chaoyun Yao, and Haoran Zhou. 2021. Gaussian mixture model clustering with incomplete data. *TOMM* 17, 1s (2021), 1–14.