Drawing CoCo Core-Sets from Incomplete Relational Data

Yongnan Liu[1,2(✉)] and Jianzhong Li[1]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
cellsi2011@gmail.com, lijzh@hit.edu.cn
[2] School of Data Science and Technology, Heilongjiang University, Harbin, China

Abstract. Incompleteness is a pervasive issue and brings challenges to answer queries with high-quality tuples. Since not all missing values can be repaired by complete values, it is crucial to provide completeness of a query answer for further decisions. To estimate such completeness results fast and objectively, CoCo core-sets are proposed in this paper. A CoCo core-set is a subset of an incomplete relational dataset, which contains tuples providing enough complete values on attributes of interest and whose ratio of complete values is close to that of the entire dataset. Based on CoCo core-sets reliable mechanisms can be designed to estimate query completeness on incomplete datasets. This paper investigates the problem of drawing CoCo core-sets on incomplete relational data. To the best of our knowledge, there is no such a proposal in the past. (1) We formalize the problem of drawing CoCo core-sets, and prove that the problem is NP-Complete. (2) An efficient approximate algorithm to draw an approximate CoCo core-set is proposed, where uniform sampling technique is employed to efficiently select tuples for coverage and completeness. (3) Analysis of the proposed approximate algorithm shows both coverage of attributes of interest and the relative error of ratio of complete attribute values between drawn tuples and the entire data can be within a given relative error bound. (4) Experiments on both real-world and synthetic datasets demonstrate that the algorithm can effectively and efficiently draw tuples preserving properties of entire datasets for query completeness estimation, and have a well scalability.

Keywords: Data quality · Data completeness · Query completeness · Incomplete data · CoCo core-sets

1 Introduction

Incomplete data are pervasive and severely reduce data quality [12,22], which brings challenges to data-driven applications [7,16] on big data. To repair missing values, data imputation [23] or data repairing methods [21] are studied from

J. Shao et al. (Eds.): APWeb-WAIM 2019, LNCS 11641, pp. 61–76, 2019.
https://doi.org/10.1007/978-3-030-26072-9_5

kinds of perspectives. However, such methods usually employ continuous time-consuming components [6,14]. Sometimes, not all missing values can be repaired, such as missing ratings for movies from users [17]. Biased query answers on incomplete data can mislead decisions [11]. To avoid such biases, the query completeness problem [19,20] is investigated to determine whether a query can get a complete answer on incomplete data under different conditions. But such methods usually suffer from high time complexity beyond NP-Completeness [13,20].

In order to determine query completeness fast, a small set of tuples with statistical properties guaranteed can be used to estimate query completeness. In such tuples from incomplete datasets, *coverage* plays an important role in providing enough information for further decisions. Therefore, to estimate query completeness fast and objectively, data used in estimation should satisfy two conditions of *coverage* and *completeness* (formally defined in Sect. 2): (1) they should provide enough kinds of information of interest; (2) the ratio of data containing different extent of information can be specified, and the ratio of complete values of chosen data is the same as that of the entire data. For instance, a movie in a recommendation system can be assigned a score by a sum of weighted ratings from different users, which can be done by a query on ratings. Since a user only watches a few movies [17], there are many missing ratings for a user. To compute the score objectively and fast, each user chosen for estimation should rate more than a specified number of movies, and such users should exceed a specified ratio. Besides, to approximately reflect the statistical properties of the entire ratings dataset, the average number of movies rated by chosen users should be almost the same as that by entire users. Therefore, to fast and objectively obtain a query completeness result, a novel data selection mechanism is crucial and should be carefully designed.

In this paper, *CoCo* core-sets with a small size satisfying the two conditions: *Coverage* and *Completeness* respectively are proposed to estimate query completeness on incomplete data fast and objectively. There may be many CoCo core-sets in the entire dataset, and any of them can provide effective and enough information. Besides, on low completeness data, CoCo core-sets can also be used as a completeness controllable subset to improve data quality for further applications, such as package queries [3] or recommendation systems [9].

Besides of the size constraint, there are other challenges to draw CoCo core-sets. (1) To satisfy the coverage condition, interesting attributes with complete values should be contained as many as possible. (2) To satisfy the completeness condition, tuples with different number of complete values should be considered. And such tuples should be carefully selected to preserve the statistical properties on the entire data. (3) The methods of drawing CoCo core-sets should be efficient and scalable.

Core-sets are designed for answering queries approximately [1]. But many methods are only tailored to numeric attribute values [1,15], which cannot be directly used in many systems containing categoric attribute values as well [10]. Furthermore, these methods cannot be used to estimate query completeness on incomplete data, since they does not consider information loss in the core-sets

brought by missing values. Methods based on sampling [2,4] can select a subset of the original dataset with statistical properties approximately guaranteed. But such subset does not necessarily satisfy the coverage condition and completeness condition. Moreover, such sampling methods may choose many samples with too many missing values on incomplete data, which make little contribution to estimation.

To overcome the drawbacks of existing methods, we propose CoCo core-sets defined formally in Sect. 2 to provide the coverage and completeness information of tuples of interest drawn from the entire dataset. The main contributions of this paper are summarized as follows:

1. CoCo core-sets used to estimate query completeness are firstly investigated, where both coverage and completeness conditions are satisfied. The problem of drawing a CoCo core-set is proved to be NP-Complete. An effective approximate algorithm to draw a high-quality CoCo core-set is proposed with linear time complexity in the size of the dataset.
2. For the completeness part, a strategy to assign error bounds is designed and proved. An algorithm based on uniform sampling using this strategy is proposed. By theoretical analysis, the tuples chosen can be proved to approximately satisfy the completeness condition at any given error bound with high probability, and the time complexity is linear in the size of the dataset.
3. For the coverage part, a strategy to guess the optimal size of tuples for coverage is proposed, followed by an efficient algorithm based on uniform sampling. By theoretical analysis, we show that our algorithm can achieve at least a $(1-1/e-\epsilon)$ approximation guarantee in expectation to the optimum coverage, and the time complexity is linear in the size of the dataset.
4. Experiments on both real-world and synthetic datasets demonstrate that the algorithm can effectively and efficiently draw tuples preserving properties of the entire dataset for query completeness estimation, and have a well scalability.

2 Problem Definition and Computational Complexity

In this section, we first introduce basic definitions of *coverage* and *completeness*, and then formalize the problem of drawing a CoCo core-set, followed by a theorem providing time complexity of the problem.

Definition 1 ***(Coverage).*** *Given a relational dataset D, the attributes $A = \{A_j\}$, $1 \leq j \leq m$, an attribute A_j is* covered *by a tuple t, if the attribute value $t[A_j]$ is complete, i.e., not missing, otherwise, A_j is* not *covered by t. The coverage of a tuples set is the size of union of attributes covered by the tuples in the set.*

Definition 2 ***(Completeness).*** *Given a relational dataset D, with tuples size n, the attributes $A = \{A_j\}$, $1 \leq j \leq m$, the* tuple completeness*, denoted as $Cpl\,(t)$, is defined as $Cpl\,(t) = \frac{\sum_{a \in A} Cpl(t[a])}{m}$, where $Cpl(t[a]) = 1$, if attribute value*

$t[a]$ is complete, otherwise $Cpl(t[a])$ is 0. The dataset completeness, *denoted as $Cpl\,(D)$, is defined as follows: $Cpl\,(D) = \frac{\sum_{t \in D} Cpl(t)}{n} = \frac{\sum_{t \in D} \sum_{a \in A} Cpl(t[a])}{mn}$, where $Cpl\,(t[a])$ is defined as above.*

Based on definitions of coverage and completeness, a CoCo core-set is defined below.

Definition 3 ***(CoCo Core-set).*** *Given a relational dataset D, the attributes $A = \{A_j\}$, $1 \leq j \leq m$, real numbers $\epsilon, \theta, \alpha \in [0, 1]$, and a positive integer y, a CoCo core-set T_{CC} is a subset of the entire dataset with minimum size satisfying:*

1. *(Property A) there are at least y attributes covered by at least one tuple;*
2. *(Property B) the completeness of the CoCo core-set is approximate to that of the entire dataset, i.e., $\left|\frac{Cpl(D) - Cpl(T_{CC})}{Cpl(D)}\right| \leq \epsilon$;*
3. *(Property C) the ratio of tuples with tuple completeness above θ is no less than α;*

In the definition above, property A corresponding to the coverage condition requires a CoCo core-set to contain complete values on several different attributes to provide enough information for further operations. The property B and property C together correspond to the completeness condition. The property B is the relative error bound property, which contributes to estimation of completeness of query answers on incomplete data, and property C makes a CoCo core-set can contain different tuples with different tuple completeness under a given ratio. In this paper, to make use of an incomplete dataset, the conflict in property C made by extreme θ is not considered.

The hardness of drawing a CoCo core-set is given below. Due to space limit, proofs are omitted. The problem can be reduced from the *Set Covering* problem with tuples constructed from given set.

Theorem 1. *Drawing a CoCo Core-set of size N or less is NP-Complete.*

3 Drawing an Approximate CoCo Core-Set

Since the problem of drawing a CoCo core-set is NP-Complete, we shall propose an efficient approximate algorithm *DCC* to draw an approximate CoCo core-set. With high probability, the coverage and completeness condition will be satisfied with error bound guaranteed. We first introduce DCC in Subsect. 3.1 and then show algorithms DCP and DCV used in DCC for completeness and coverage in Subsects. 3.2 and 3.3 respectively.

3.1 An Approximate Algorithm

Intuitively, since a CoCo core-set consists of tuples for two goals: completeness and coverage, DCC will draw tuples by two steps. Firstly, tuples for completeness

are drawn to make sure the dataset completeness of such drawn tuples is close to that of the entire tuples. Secondly, tuples for coverage are drawn to cover as many uncovered attributes as possible. Such uncovered attributes are not covered by the tuples for completeness. Since DCC usually cannot know the optimal size of tuples covering all specified attributes, DCC guesses the optimal size and then selects tuples for coverage by a greedy mechanism. After these two steps, tuples for completeness and coverage are combined together as an approximate CoCo core-set. To make the size of an approximate CoCo core-set as small as possible, the numbers of tuples for completeness and coverage are carefully designed.

Algorithm 1. DCC

Input : a relational dataset D with n tuples and attributes of size m, real numbers $\epsilon_{CV}, \epsilon_{CP}, \delta, \theta, \alpha > 0$, and a positive integer y
Output: An approximate CoCo core-set T_{CC}

1: $T_{CC} \leftarrow \phi$;
`// Drawing tuples for completeness`
2: $T_{CP} \leftarrow$ `DCP`$(D, \epsilon_{CP}, \delta, \theta, \alpha)$;
`// Drawing tuples for coverage`
3: $T_{CV} \leftarrow$ `DCV`$(D \backslash T_{CP}, T_{CP}, \epsilon_{CV}, y)$;
`// Combining tuples`
4: $T_{CC} \leftarrow T_{CV} \cup T_{CP}$;
5: **return** T_{CC};

The algorithm DCC is shown in Algorithm 1. As the intuitive idea, an approximate CoCo core-set consists of the tuples for completeness T_{CP} drawn by DCP and the tuples for coverage T_{CV} drawn by DCV. Given a dataset with bounded number of attributes, since the time complexity of DCP and DCV is $O(n)$ and $O(n)$ respectively shown next, the time complexity of DCC is $O(n)$, where n is the number of the tuples. Note that, the size of the approximate CoCo core-set drawn by DCC is determined by the given error bounds for completeness and coverage.

3.2 Drawing Tuples for Completeness

To select tuples satisfying Property B in Definition 3 efficiently, methods such as [5] based on uniform sampling are proposed. Tailored to complete dataset, such a method cannot provide samples satisfying Property C in Definition 3.

To draw tuples for completeness efficiently, DCP performs uniform sampling to draw tuples. Given a relative error bound ϵ and failure probability δ, the minimum sample size (MSS) is given in Eq. (1) [5], where $\phi_{\delta/2}$ is the $\delta/2$ fractile of the standard normal distribution, and $\inf(C_t)$ and $\sup(C_t)$ are the minimum and maximum number of complete values in a tuple respectively. Therefore, the

minimum sample size is determined by error bound ϵ and failure probability δ.

$$MSS(\epsilon, \delta) = \left\lceil \frac{\phi^2_{\delta/2}}{\epsilon^2} \left(\frac{\sup(C_t)}{\inf(C_t)} - 1 \right) \right\rceil \tag{1}$$

To satisfy Property C, methods to determine sizes of tuples with different tuple completeness must be carefully designed. When tuples from different parts are combined, the dataset completeness of combined samples should be close to that of the entire original dataset. To achieve this goal, the given error bound and failure probability must be divided so that sizes of tuples with different tuple completeness can be determined according to corresponding error bounds and failure probabilities.

Next, we first introduce a strategy to assign error bound and failure probability to tuples with different tuples completeness, and then DCP is proposed based on this strategy.

Assigning Errors and Failure Probabilities. When a real number θ is given, tuples are divided into two parts by its tuple completeness. One is tuples with tuples completeness at least θ, denoted as T_H, and the other is tuples with tuples completeness less than θ, denoted as T_L. Clearly, the dataset completeness of the entire data is given as below:

$$Cpl(D) = \rho_H \times \frac{|T_H|}{|D|} + \rho_L \times \frac{|T_L|}{|D|}$$

where $\rho_H = Cpl(T_H)$ and $\rho_L = Cpl(T_L)$. If ρ_H and ρ_L can be approximated by sampling, then $Cpl(D)$ can be approximated by sampling. Therefore, with different errors and failure probabilities, sample sizes of T_H and T_L can be determined.

To describe how well a random variable is approximated, (ϵ, δ) - approximation is defined in Definition 4 [5].

Definition 4. *((ϵ, δ) - approximation) $\hat{Y}$ is called as an (ϵ, δ) - approximation of Y if $P\left(\left|\frac{\hat{Y}-Y}{Y}\right| \geq \epsilon\right) \leq \delta$ for any $\epsilon \geq 0$ and $0 \leq \delta \leq 1$, where $P(X)$ is the probability of a random variable X.*

Now we introduce a strategy to assign error bounds and failure probabilities between samples from T_H and T_L.

Theorem 2. *$\widehat{Cpl(D)}$ is an (ϵ, δ) - approximation of $Cpl(D)$, if $Cpl(S_H)$ is an (ϵ, Δ_H) - approximation of $Cpl(T_H)$, and $Cpl(S_L)$ is an (ϵ, Δ_L) - approximation of $Cpl(T_L)$, satisfying $\Delta_H + \Delta_L \leq \delta$, and sizes of S_H and S_L are given by Eq. (1).*

By Theorem 2, tuples with different tuple completeness can be sampled differently, according to different goals. For T_H, the error bound is ϵ_H, and the failure probability Δ_H. And ϵ_L and Δ_L are the counterparts for T_L. From Theorem 2, we can set $\epsilon_H = \epsilon_L = \epsilon$, and $\Delta_H + \Delta_L = \delta$. By Eq. (1), since given a fixed

Algorithm 2. DCP

Input : a relational dataset D with n tuples, real numbers $\epsilon > 0$, $\delta > 0$, $\theta > 0$, $\alpha > 0$

Output: a tuples set T_{CP} for completeness

$T_{CP} \leftarrow \phi$;
// compute error probabilities of each part
$\Delta \leftarrow$ AssignPros$(\epsilon, \delta, \alpha)$; // Δ_H and Δ_L are both in Δ
// sampling on tuples with tuple completeness at least θ
$s_H \leftarrow$ MSS(ϵ, Δ_H);
$S_H \leftarrow$ UniformSampling(D, s_H);
$T_{CP} \leftarrow T_{CP} \cup S_H$;
// sampling on tuples with tuple completeness under θ
$s_L \leftarrow$ MSS(ϵ, Δ_L);
$S_L \leftarrow$ UniformSampling(D, s_L);
$T_{CP} \leftarrow T_{CP} \cup S_L$;
return T_{CP};

ϵ, a sample size is determined monotonely with δ, proper sample sizes satisfying Property C can be obtained by binary search on δ.

The Algorithm DCP. The algorithm DCP based on uniform sampling is shown in Algorithm 2.

To describe DCP consistently, symbols defined above are continuously used. As is shown above, given error bound ϵ, failure probability δ, and ratio requirement α, the failure probability Δ_H and Δ_L can be determined. With ϵ and Δ_H, sample size s_H for uniform sampling on T_H can be computed by Eq. (1), where $\sup(C_t)$ can be set to m, and $\inf(C_t)$ can be set to 1. Then by uniform sampling implemented in a reservoir [24], a tuples set S_H is sampled and added into T_{CP}. And then uniform sampling on T_L is similar. Therefore, T_{CP} contains tuples satisfying Property B and Property C in Definition 3, which makes completeness condition is approximately satisfied. Since there are 2 scans on T_H and T_L to sample tuples, and either scan takes $O(n)$ by using reservoir sampling, the time complexity of DCP is $O(n)$, where n is the number of the tuples.

3.3 Drawing Tuples for Coverage

Since the coverage function defined on a tuple set is submodular, to efficiently draw tuples for coverage, sampling is a feasible methodology to obtain tuples of interest with controllable error bound and high probability. Like the sampling method stochastic-greedy [18] or SG for short, DCV uses a similar idea of greedily covering attributes by choosing a best sampled tuple covering most uncovered attributes in each round. For SG, the number of tuples k for coverage is fixed by a user. Therefore, SG tries to use k tuples to cover as many attributes as possible. But for DCV, the optimal size of tuples for covering uncovered attributes is usually unknown. What's worse, a user cannot provide such a size either. To get a proper trade-off between coverage and size, DCV shown below tries to

Algorithm 3. DCV

Input : a relational dataset D with n tuples, a tuples set T_{CP}, a real number $\epsilon > 0$, and a positive integer y
Output: a tuples set T_{CV} for coverage

1: $T_{CV} \leftarrow \phi$;
2: $l \leftarrow \max(\lfloor \ln \frac{1}{\epsilon} \rfloor, 1)$;
// Obtain the number of not covered attributes
3: $h \leftarrow$ Uncovered(T_{CP}, y);
// Guessing the number of optimal size in each round
4: $k_G \leftarrow \frac{h}{l}$;
// cover attributes by l rounds
5: **for** $i \leftarrow 1$ **to** l **do**
    // Sample s tuples by uniform sampling
6: $s \leftarrow \frac{n}{k_G} \ln \frac{1}{\epsilon}$;
7: **for** $j \leftarrow 1$ **to** k_G **do**
8: $Samples \leftarrow$ UniformSampling(D, s);
        // Find the tuple covering most uncovered attributes
9: $t_j \leftarrow$ FindBest$(Samples)$;
10: $T_{CV} \leftarrow T_{CV} \cup \{t_j\}$;
11: **return** T_{CV};

guess a proper size, and then DCV uses SG multiple times to cover attributes. With theoretical analysis below, we can show that DCV covers more uncovered attributes, with at most a constant factor more samples.

The Algorithm DCV. The algorithm DCV based on uniform sampling is shown in Algorithm 3. Given a user specified number y of attributes to cover, DCV needs to know which attributes are not covered by the tuples in T_{CP}. And then DCV uses l rounds to cover these attributes. To take advantage of SG, DCV guesses a proper size k_G, which is size of tuples chosen in each round. With k_G, sample size for uniform sampling in each round can be computed. Among the samples, a tuple covering most uncovered attributes is selected and added into the result set. DCV can be easily modified to stop earlier than k_G rounds as soon as all attributes are covered.

There is an alternative method, denoted as DSG. Since each tuple can at least cover one uncovered attributes, we need at most h tuples. Therefore, after knowing that which attributes are not covered, DSG can simply invoke SG to cover such attributes with $k = h$. With fewer samples in each round, DSG is faster than DCV. But as the theoretical results shown below and experimental results shown in Sect. 4, DSG usually selects almost the same size of best tuples as DCV, but obtains low coverage due to a smaller samples size in each sampling.

In each uniform sampling, by using a reservoir [24], it takes $O(n)$ to obtain samples, where $n = |D|$. It takes $O(s)$ to find a best tuple to cover most remaining uncovered attributes, given a small or bounded number of attributes. The time complexity of DCV is $O(n)$, where $n = |D|$.

Since DCV usually cannot know the optimal size of tuples for covering uncovered attributes, DCV has to guess a size of tuples to cover these attributes. With such guessed size, DCV runs multiple times of SG to efficiently cover attributes as Lemma 1. The coverage and time complexity of DCV is shown in Theorem 3.

Lemma 1. *Given a size k of tuples for covering attributes, after l times run of SG, the coverage in expectation is at least $f(I^*)\left(1 - l\epsilon - e^{-l}\right)$, where $f(I^*)$ is the optimum coverage with such k tuples. And if $l \leq \ln \frac{1}{\epsilon}$, the more times SG is run, the more coverage in expectation is obtained.*

From Lemma 1, though multiple runs of SG improve coverage, too many times run may not bring more benefit. Therefore, DCV guess a proper size of tuples for covering, if such guessed size is too small, DCV can take advantage of multiple runs to obtain great improvement. Otherwise, since each tuple can at least cover one attribute, the optimal size is at most h, which makes DCV achieve the same approximate ratio as SG. Based on this idea, the coverage of DCV is shown in Theorem 3.

Theorem 3. *If l is set to $l = \ln \frac{1}{\epsilon}$, and k_G is set to $k_G = \frac{h}{l}$, then with $O(n(\ln \frac{1}{\epsilon})^2)$ coverage function evaluations, DCV achieves a $(1-\epsilon+\epsilon \ln \epsilon)$ approximation guarantee in expectation to the optimum solution if $k_G \geq k^*$, and if $k_G < k^*$, DCV achieves a $(1 - \epsilon - 1/e)$ approximation, where h is the number of uncovered attributes and k^* is the optimal size of tuples covering such attributes.*

4 Experimental Results

In this section, we will show the experimental results of the proposed algorithm DCC on different datasets with different parameters. All the experiments are implemented on a Microsoft Windows 10 machine with an Intel(R) Core i7-6700 CPU 3.4 GHz and 32 GB main memory. Programs are compiled by Microsoft Visual Studio 2017 by C++ language. Each program is run 10 times on the same dataset to show stability, and the time cost given below is the average time cost. The completeness is defined as in Sect. 2.

4.1 Datasets and Metrics

MovieLens Dataset. The relation MovieLens (ML for short) is downloaded from MovieLens website[1], which is marked as MovieLens 1M Dataset on the website. Specially, our MovieLens contains 6040 records (users ratings, varying from 1 to 5), with 3952 dimensions (movies), and the size is 89.1 MB. The missing ratio is 95.81%, namely, including 1 million valid ratings. Actually, there are 246 movies without any ratings. Though there are many attributes (movies)

[1] MovieLens dataset: https://grouplens.org/datasets/movielens/1m/.

compared to records, we can still view the number of attributes as a bounded number, since such data is a part of the whole dataset, and usually there are far more users than movies.

Cars Information. The relation *Cars* is crawled from a website[2] providing services on cars. Without changing any value, we built the relation Cars including information from dealers selling new cars. The relation Cars consists of 8 attributes of a dealer: name, stock, star-rate, ratings, location, distance from the main company, phone number, and email. There are 542850 tuples in this relation, and the size is 152 MB. The dataset completeness of the relation Cars is 0.843.

Synthetic Relations. To evaluate efficiency of our algorithm DCC, we generated a collection of synthetic relations with 100 attributes and tuples number from 10K to 5M, whose sizes are from 5.38 MB to 2.65 GB correspondingly. The synthetic datasets are generated in two steps. First, there are 100 tuples with only one complete attribute value in each tuple to cover all the attributes, which is used to test effectiveness of algorithms to cover attributes. Second, there are other tuples with incomplete attribute values, where attribute values are missing with different probabilities. Detailed information are shown in Subsect. 4.3.

Metrics. We shall evaluate our algorithm by four metrics: relative error, coverage, efficiency and size of the approximate CoCo core-set. *Relative Error* denoted as *RE* measures the extent to which the approximate CoCo core-set can give an estimate of the dataset completeness of the entire dataset, which is shown by the relative error between the estimated completeness by the approximate CoCo core-set and the real dataset completeness. *Coverage* measures the number of attributes of interest covered. *Efficiency* measures the time cost of an algorithm, and *Size* measures the size of the approximate CoCo core-set drawn by our algorithm DCC. To show coverage improvement, we also implemented stochastic-greedy or SG for short, from [18] in C++.

According to Algorithm 1, DCC can be further evaluated by performance of DCP and DCV respectively. The performance of DCP is evaluated by *RE*, *size* and *efficiency* under different error bounds ϵ_{CP}, while DCV is evaluated by *size*, *coverage* and *efficiency* under different error bounds ϵ_{CV}. In all the experiments below, y is set be the attributes number of the dataset. We shall first report experimental results on real datasets, and then we shall give results on synthetic datasets used to further evaluate DCC.

[2] Cars dataset: http://www.cars.com.

Table 1. Experimental results of DCP on real datasets

ϵ_{CP}	MovieLens			Cars		
	RE	Size	TimeCost (s)	RE	Size	TimeCost (s)
0.10	0.003	3943	11.867	0.019	1875	316.297
0.15	0.006	2803	11.443	0.018	833	317.432
0.20	0.002	1699	11.250	0.017	468	317.680
0.25	0.003	1088	11.014	0.018	299	315.806
0.30	0.021	754	11.178	0.018	207	313.929

4.2 Experiments on Real Datasets

Performance of DCP. To investigate performance of DCP, we ran a group of experiments to observe RE, sizes and efficiency under different ϵ_{CP}, and the results are shown in Table 1 and depicted in Fig. 1. We set $\epsilon_{CV} = 0.05$, the failure probability $\delta_{CP} = 0.6$, and θ is dataset completeness of such two real datasets, $\alpha = 0.4$. For T_H, we set $\Delta_H = 0.2$, and $\Delta_L = 0.4$ for T_L. Notice that we set a large failure probability δ_{CP}, because incomplete dataset can lead to a large failure probability. But since we can assign failure probabilities, we set Δ_H to be smaller, because we want more tuples with large tuple completeness to reduce failure probability on the entire dataset, which is demonstrated true as the experimental results shown below. The variable $sup(C_t)$ in Eq. (1) on the MovieLens dataset is set to be 30, since there are too many missing values, while $Sup(C_t)$ is set to be 9 on the Cars dataset, since tuples in Cars dataset are with high tuple completeness and we add an extra attribute ID. The variable $inf(C_t)$ in both MovieLens and Cars dataset are set to be 1, since there is at least one attribute without missing values.

Relative Errors of DCP. As in Table 1, real relative errors in the RE columns under both datasets are very small, far from the corresponding given relative error bounds ϵ_{CP}, which shows that the strategy to assign relative errors and failure probabilities is effective. And tuples in T_{CP} can provide good estimates of the real dataset completeness of the two real datasets. For the MovieLens dataset, there are many tuples with low tuple completeness, which leads to a low dataset completeness. Since T_{CP} can give good estimates under different relative error bounds, a user can use T_{CP} to analyze the query completeness on such low completeness dataset, which will give guides for designing queries with completeness guaranteed.

Sizes in DCP. In Fig. 1, on the MovieLens dataset, tuples sampled with high tuple completeness no less than θ is denoted as H-ML, while tuples sampled with low tuple completeness less than θ is denoted as L-ML, and the sum of the two parts are denoted as All-ML. H-Cars, L-Cars and All-Cars are defined similarly on the Cars dataset. Figure 1 depicts the different sizes of tuples in T_{CP} on the two real datasets with different ϵ_{CP}.

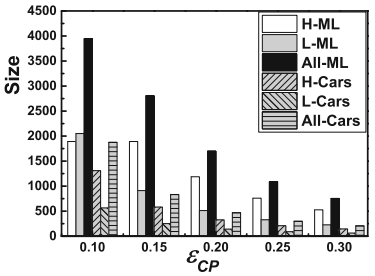

Fig. 1. Sizes of T_{CP} on real datasets

As is shown in Fig. 1, Algorithm 2 samples different sizes of tuples with different tuple completeness according to given θ and α by Theorem 2. And the completeness condition is satisfied on all the experiments. For example, when $\epsilon_{CP} = 0.2$, there are 327 tuples sampled with high tuple completeness, and 141 tuples sampled with low tuple completeness on the Cars dataset. On the MovieLens dataset, there are 1187 tuples with high tuple completeness and 512 tuples low tuple completeness. Since there are only 1892 tuples with tuple completeness no less than given θ, when $\epsilon_{CP} = 0.10$, all the 1892 tuples are sampled. To get a good estimate of $Cpl(ML)$, Algorithm 2 has to select 2051 tuples. On the Cars dataset, since there are sufficient tuples with different tuple completeness, Algorithm 2 always selects more tuples on high tuple completeness. Therefore, by Theorem 2, tuples with different tuple completeness can be drawn to improve data usability from low-quality datasets according to given ratio requirement for further use.

Efficiency of DCP. As is shown in Table 1, changes of ϵ_{CP} do not bring obvious changes of time cost on both datasets. Since the uniform sampling is implemented by reservoir sampling [24], which needs to scan all the dataset. Therefore, given the same dataset, the time cost stays almost the same.

Table 2. Experimental results of DSG and DCV on the MovieLens dataset

ϵ_{CV}	DSG				DCV			
	TimeCost (s)	Size	Coverage	ToCover	TimeCost (s)	Size	Coverage	ToCover
0.05	293.383	107	169	178	296.432	115	175	178
0.10	282.515	96	154	172	285.697	106	168	172
0.15	393.939	124	221	239	392.969	119	219	239
0.20	298.532	98	150	181	297.446	104	155	181
0.25	417.606	138	200	252	415.157	119	226	252
0.30	316.068	93	157	190	313.560	90	155	190

Performance of DCV. To investigate performance of DCV, we ran a group of experiments to observe coverage and efficiency with different tuple sizes under different ϵ_{CV}, and the results are shown in Table 2. In the experiments, we set $\epsilon_{CP} = 0.3$, failure probability $\delta_{CP} = 0.6$, and $\alpha = 0.4$. For T_H, we set $\Delta_H = 0.2$, and $\Delta_L = 0.4$ for T_L. The variable $sup(C_t) = 30$ on the ML dataset, and $sup(C_t)$ is set to be 9 on the Cars dataset. The variable $inf(C_t) = 1$ in both ML and Cars dataset. Since DCV uses the similar idea of SG [18], to show the improvement of DCV, we modified the Algorithm 1 into a new Algorithm DSG, which invokes SG after invoking DCP on the same dataset used in DCC. On the Cars dataset, since there are many tuples with high tuple completeness, T_{CP} always covers all the attributes, which makes Algorithm 3 skipped. Therefore, only experimental results of DCV on the ML dataset is shown below. In Table 2, experimental results of the two algorithms: DSG and DCV are shown separately. For each ϵ_{CV}, DCV was run 10 times, the worst coverage is listed in the column *Coverage*, with the time cost in seconds in the column *TimeCost(s)*, size of T_{CV} in the column *Size*, and number of uncovered attributes by T_{CP} in the column *ToCover*. And the corresponding results are shown in the counterparts under the *DSG* part.

Coverage of DCV. On the MovieLens dataset, since there are a few ratings, which brings challenges to Algorithm 3, but as shown in Table 2, both DSG and DCV covers enough attributes under different ϵ_{CV} with as small size of T_{CV} as possible. As is shown in Table 2, when $\epsilon_{CV} \leq 0.2$, DCV covers most attributes, and covers more attributes than DSG, even if some attributes are with lower frequencies. Because a samller ϵ_{CV} leads to a larger l, which brings more chances to cover attributes as in Theorem 3. When ϵ_{CV} is too large, such as $\epsilon_{CV} \geq 0.4$, l can be very small. Since l is at least 1, which makes DCV perform almost the same as DSG.

Efficiency of DCV. As shown in Table 2, with almost the same time as DSG, DCV covers most uncovered attributes. Since there are only 6040 tuples in the MovieLens, and reservoir sampling is used, multiple tries of sampling the best tuples for coverage do not bring too much time cost with different ϵ_{CV}. To evaluate DCV more deeply, synthetic relations are generated based on the MovieLens dataset, and the experimental results are shown in Subsect. 4.3.

4.3 Experiments on Synthetic Relations

To further evaluate our algorithms, we generated synthetic relations with different dataset completeness. Besides, rare attributes with fewer frequencies were generated to test coverage of DCV. The results are shown in Table 3 and Fig. 2, where the results are the average values of 10 runs under each number of tuples, and *TC* is short for Time Cost in seconds. In the experiments, for coverage, we set $\epsilon_{CV} = 0.1$, θ is the dataset completeness of current dataset. For completeness, we set $\epsilon_{CP} = 0.3$, $\Delta_H = 0.2$, $\Delta_L = 0.4$, and $\delta = 0.6$.

As is shown in Table 3, though we set a loose error bound and high failure probability in DCP, DCC can still obtain small relative errors as Theorem 2

Table 3. Experimental results on synthetic relations

#Tuples	Size	TC (s)	RE	Coverage
10k	2640	14.319	0.017	92
50k	2636	69.278	0.008	94
100k	2648	161.424	0.003	96
500k	2639	686.560	0.005	95
1M	2647	1576.560	0.003	94
5M	2641	7024.790	0.006	93

shows. And the size of approximate CoCo core-set is rather small compared to the entire dataset. For example, there are 1M tuples in the entire dataset, while there are only 2647 tuples in the CoCo core-set with relative error 0.00294 and 94 attributes covered on average.

As depicted in Fig. 2, where both number of tuples and time cost are shown in logarithm with base 10, the algorithm DCP is linear as analyzed. Though DCV is a little worse than linearity, it does not bring too much time to DCC, which shows well scalability of our algorithms.

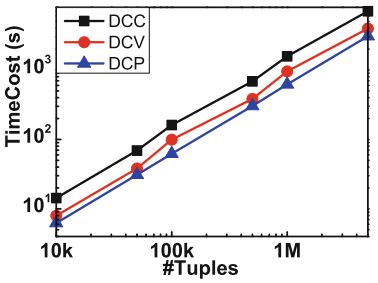

Fig. 2. Time cost on synthetic relations

5 Related Work

Core-sets for Numeric Values. The concept of numeric core-sets is firstly from [1], followed by many other methods such as [4,15]. There are two main drawbacks of methods above. One is that they are only numeric core-sets, and the other is that only aggregated values are stored, which leads to loss of detailed features described in the origin dataset.

Drawing Core-sets by Sampling. Sampling can be used to draw a subset of original entire data, such as [2,8]. In the papers above, the minimized sample sizes can be computed in that each sample on complete data can make contributions to

approximately answering queries. But due to missing values on incomplete data, effective information may be lost in samples, which makes it hard to provide minimized sample sizes for different queries. Tuples containing different extent of information should be sampled differently.

Determining Query Completeness. The problems of whether a query can get a complete answer on different kinds of data are studied by many researchers [13,20]. Such problems can be called *query completeness* for short. The time complexity of such a problem is usually beyond NP-Complete, which makes it very hard to determine query completeness fast on the entire dataset. A CoCo core-set with a small size is drawn from the entire dataset, which makes it possible to design efficient methods to estimate query completeness fast based on a CoCo core-set.

6 Conclusion

This paper investigates the problem of drawing CoCo core-sets on incomplete relational data. We analyze the time complexity of the problem and prove that it is NP-Complete. To solve this problem effectively and efficiently, an algorithm is proposed to draw an approximate CoCo core-set based on uniform sampling techniques. We prove that our algorithm can draw an approximate CoCo core-set, which almost preserves coverage and completeness properties of the entire dataset. Experimental results on both real-world and synthetic datasets show that our algorithms can draw an effective CoCo core-set on low-quality dataset as theoretical analysis, and our algorithms have a well scalability on larger datasets. Based on an approximate CoCo core-set, query completeness can be estimated fast and objectively, and the algorithms of estimation are our future work.

Acknowledgment. This work was supported by the National Key R&D Program of China under Grant 2018YFB1004700, and the National Natural Science Foundation of China under grants 61832003, U1811461 and 61732003.

References

1. Agarwal, P.K., Har-Peled, S., Varadarajan, K.R.: Approximating extent measures of points. J. ACM **51**(4), 606–635 (2004)
2. Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S., Stoica, I.: BlinkDB: queries with bounded errors and bounded response times on very large data. In: Proceedings of the 8th ACM European Conference on Computer Systems, pp. 29–42. ACM (2013)
3. Brucato, M., Abouzied, A., Meliou, A.: Package queries: efficient and scalable computation of high-order constraints. VLDB J. **27**, 1–26 (2017)
4. Cheng, S., Cai, Z., Li, J., Fang, X.: Drawing dominant dataset from big sensory data in wireless sensor networks. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 531–539. IEEE (2015)
5. Cheng, S., Li, J.: Sampling based (ϵ, δ)-approximate aggregation algorithm in sensor networks. In: 29th IEEE International Conference on Distributed Computing Systems, ICDCS 2009, pp. 273–280. IEEE (2009)

6. Chu, X., et al.: Katara: a data cleaning system powered by knowledge bases and crowdsourcing. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1247–1261. ACM (2015)
7. Chung, Y., Krishnan, S., Kraska, T.: A data quality metric (DQM): how to estimate the number of undetected errors in data sets. Proc. VLDB Endow. **10**(10), 1094–1105 (2017)
8. Cohen, E., Kaplan, H.: Tighter estimation using bottom k sketches. Proc. VLDB Endow. **1**(1), 213–224 (2008)
9. Deng, T., Fan, W., Geerts, F.: On the complexity of package recommendation problems. In: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2012, pp. 261–272 (2012)
10. Deng, T., Fan, W., Geerts, F.: On recommendation problems beyond points of interest. Inf. Syst. **48**, 64–88 (2015)
11. Dong, X.L., et al.: Knowledge-based trust: estimating the trustworthiness of web sources. Proc. VLDB Endow. **8**(9), 938–949 (2015)
12. Fan, W.: Data quality: from theory to practice. ACM SIGMOD Rec. **44**(3), 7–18 (2015)
13. Fan, W., Geerts, F.: Capturing missing tuples and missing values. In: Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 169–178. ACM, June 2010
14. Hao, S., Tang, N., Li, G., Li, J.: Cleaning relations using knowledge bases. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 933–944. IEEE (2017)
15. Indyk, P., Mahabadi, S., Mahdian, M., Mirrokni, V.S.: Composable core-sets for diversity and coverage maximization. In: Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2014, pp. 100–108. ACM (2014)
16. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline query processing for incomplete data. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 556–565. IEEE (2008)
17. Miao, X., Gao, Y., Zheng, B., Chen, G., Cui, H.: Top-k dominating queries on incomplete data. IEEE Trans. Knowl. Data Eng. **28**(1), 252–266 (2016)
18. Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrak, J., Krause, A.: Lazier than lazy greedy. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
19. Razniewski, S., Korn, F., Nutt, W., Srivastava, D.: Identifying the extent of completeness of query answers over partially complete databases. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 561–576. ACM (2015)
20. Razniewski, S., Nutt, W.: Completeness of queries over incomplete databases. Proc. VLDB Endow. **4**(11), 749–760 (2011)
21. Rekatsinas, T., Chu, X., Ilyas, I.F., Ré, C.: Holoclean: holistic data repairs with probabilistic inference. Proc. VLDB Endow. **10**(11), 1190–1201 (2017)
22. Sadiq, S., et al.: Data quality: the role of empiricism. SIGMOD Rec. **46**(4), 35–43 (2018)
23. Song, S., Zhang, A., Chen, L., Wang, J.: Enriching data imputation with extensive similarity neighbors. Proc. VLDB Endow. **8**(11), 1286–1297 (2015)
24. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. (TOMS) **11**(1), 37–57 (1985)