Data & Knowledge Engineering 87 (2013) 146–166

Contents lists available at ScienceDirect

Data & Knowledge Engineering

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

Editorial

Efficient discovery of similarity constraints for matching dependencies

Shaoxu Song [a,b,*], Lei Chen [c]

[a] Key Laboratory for Information System Security, MOE, School of Software, Tsinghua University, China
[b] TNList, School of Software, Tsinghua University, China
[c] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 15 December 2011
Received in revised form 12 June 2013
Accepted 12 June 2013
Available online 29 June 2013

Keywords:
Data dependencies
Matching dependencies
Management of integrity constraints
Database integration

ABSTRACT

The concept of *matching dependencies* (MDs) has recently been proposed for specifying matching rules for object identification. Similar to the functional dependencies (with conditions), MDs can also be applied to various data quality applications such as detecting the violations of integrity constraints. In this paper, we study the problem of discovering similarity constraints for matching dependencies from a given database instance. First, we introduce the measures, support and confidence, for evaluating the utility of MDs in the given data. Then, we study the discovery of MDs with certain utility requirements of support and confidence. Exact algorithms are developed, together with pruning strategies to improve the time performance. Since the exact algorithm has to traverse all the data during the computation, we propose an approximate solution which only uses part of the data. A bound of relative errors introduced by the approximation is also developed. Finally, our experimental evaluation demonstrates the efficiency of the proposed methods.

1. Introduction

Recently, data quality has become a hot topic in the database community due to the huge amount of "dirty" data originated from different resources (see [4] for a survey). These data often contain duplicates, inconsistencies and conflicts, due to various errors introduced by human and machines [35]. In addition to the cost of dealing with the huge volume of data, manually detecting and removing "dirty" data is definitely out of practice because human proposed cleaning methods may introduce inconsistencies again. Therefore, data dependencies, which have been widely used in the relational database design to set up the integrity constraints, have been revisited and revised to capture wider inconsistencies in the data [16,12,33,45]. For example, consider a Contacts relation with the schema:

$$\text{Contacts}(\text{SIN}, \text{Name}, \text{CC}, \text{ZIP}, \text{City}, \text{Street}).$$

The following functional dependency fd specifies a constraint that for any two tuples in Contacts, if they have the same ZIP code, then these two tuples have the same City as well. Recently, *functional dependencies* (FDs) have been extended to *conditional functional dependencies* (CFDs) [8], i.e., FDs with conditions, which have more expressive power. The basic idea of these extensions is making the FDs, that originally hold for the whole table, valid only for a set of tuples. For example, the following cfd specifies that

* Corresponding author at: TNList, School of Software, Tsinghua University, China. Tel.: +86 10 62795435.
E-mail addresses: sxsong@tsinghua.edu.cn (S. Song), leichen@cse.ust.hk (L. Chen).

0169-023X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.datak.2013.06.003

only in the condition of country code CC = 44, if two tuples have the same ZIP, then they must have the same Street as well.

$$\begin{aligned} \text{fd} &: [\text{ZIP}] \rightarrow [\text{City}] \\ \text{cfd} &: [\text{ZIP}, \text{CC} = 44] \rightarrow [\text{Street}] \end{aligned}.$$

These dependency constraints can be used to detect data integrity violations [14]. For instance, we can use the above fd to detect violations in an instance of Contacts in Table 1. The tuples t_5 and t_6 with the same values of ZIP = 021 have different values of City. Thus, these tuples are detected as violations of the above fd.

Although functional dependencies (and their extension with conditions) are very useful in determining data inconsistency and repairing the "dirty" data [14], they check the specified attribute value agreement based on the *equality*. For example, with the above cfd, tuples that have CC = 44 and the same value on ZIP attribute will be checked to see whether they have exactly equal values on Street. Obviously, this strict equality constraint limits the usage of FDs and CFDs, since real-world information often has various representation formats. For instance, the tuples t_2 and t_3 in the Contacts table will be detected as "violations" of the cfd, since they have "different" Street values but agree on ZIP and CC = 44. However, "No.2, Central Rd." and "#2, Central Rd." are indeed the "same" street in the real-world with different representation formats.

To make dependencies adapt to this real-world scenario, i.e., to be tolerant of various representation formats, Fan [16] proposed a new concept of data dependencies, called *matching dependencies* (MDs). Informally, a matching dependency targets on the fuzzy values like text attributes. It specifies the dependency between two set of attributes according to their matching quality measured by some similarity matching operators, such as *Euclidean distance* and *cosine similarity* (see [6] for a survey). Again, in the Contacts example, we may have an MD as

$$\text{md}_1 : ([\text{Street}] \rightarrow [\text{City}], <0.8,\ 0.7>)$$

which states that for any two tuples from Contacts, if they agree on attribute Street (the matching similarity, e.g. *cosine similarity*, on the attribute Street is greater than a threshold 0.8), then the corresponding City attribute should match as well (i.e. similarity on City is greater than the corresponding threshold 0.7). Consequently, we can identify the violation between t_5 and t_6, since they share the same Street, i.e., with cosine similarity 1.0 greater than 0.8, but have City value similarity 0.0 less than the threshold 0.7.

Similar to the FDs related techniques, MDs can be applied in many data quality tasks as well [16]. First, in data cleaning, we can also use MDs to detect the inconsistent data, i.e., the data that do not follow the constraint (rule) specified by MDs. For example, according to the above md_1, for any two tuples t_i and t_j having similarity greater than 0.8 on Street, they should be matched on City as well (similarity $\geq$ 0.7). If their City similarity is less than 0.7, then there must be something wrong in t_i and t_j, i.e., inconsistency. Such inconsistency on text attributes cannot be detected by using FDs (and the extensions) based on the equality. In addition to locating the inconsistent data, object identification, another important work for data cleaning, can also employ MDs as matching rules [19]. For instance, according to

$$\text{md}_2 : ([\text{Name},\ \text{Street}] \rightarrow [\text{SIN}],\ <0.9,\ 0.9,\ 1.0>)$$

if two tuples have high similarities on Name and Street (both similarities are greater than 0.9), then these two tuples probably denote the same person in the real world, i.e., having the same SIN. Note that the tuple pair (t_5,t_6) is not a violation to md_2, since their similarity on attribute Name is 0.7 which is less than 0.9. The dependency md_2 only evaluates the tuple pair which has high similarities on Name and Street (i.e., both similarities are greater than 0.9).

1.1. Motivation

Matching dependencies have already been proved to be useful in applications [19,17]. In this paper, we focus on how to discover such useful MDs. In fact, given a database instance, there are enormous MDs that can be discovered if we set different similarity thresholds on attributes. Note that if all thresholds are set to 1.0, i.e., the equality case, MDs have the same semantics as traditional FDs. In other words, traditional FDs are special cases of MDs. For instance, the above fd can be represented by an MD ([ZIP] → [City], <1.0, 1.0>). Clearly, not all the settings of thresholds for MDs are useful.

Table 1
Example of Contacts relation $\mathcal{R}$.

SIN	Name	CC	ZIP	City	Street	
584	Claire Green	44	606	Chicago	No.2, Central Rd.	t_1
584	Claire Greem	44	606	Chicago	No.2, Central Rd.	t_2
584	Claire Gree	44	606	Chicago	#2, Central Rd.	t_3
265	Jason Smith	01	021	Boston	No.3, Central Rd.	t_4
265	J. Smith	01	021	Boston	#3, Central Rd.	t_5
939	W. J. Smith	01	021	Chicago	#3, Central Rd.	t_6

Following [11], we employ the *confidence* and *support* measures to evaluate the utility of MDs in the above applications. Specifically, let's consider an MD of a relation $\mathcal{R}$, denoted by $\varphi(X \rightarrow Y, \lambda)$, where X and Y are the attribute sets of $\mathcal{R}$, λ is a pattern specifying different similarity thresholds on each attribute in X and Y. Let λ_X and λ_Y be the projections of thresholds in pattern λ on the attributes X and Y respectively.

- The *support* of φ is the proportion of tuple pairs whose matching similarities are higher than the thresholds in φ on both attributes of X and Y.
- The *confidence* is the ratio of tuple pairs whose matching similarities satisfy λ_X also satisfying λ_Y.

In real applications inconsistency detection, in order to achieve high detection accuracy, we would like to use MDs with high confidence. Otherwise, if users need high recall of object identification, then MDs with high support are preferred. Intuitively, it is desirable to discover those MDs with high support and high confidence.

In this work, we study the problem of discovering proper settings of similarity thresholds for MDs, which can satisfy users' utility requirements of support and confidence. It is worth noting that similarity constraints are not an issue in the previous research on the discovery of FDs [28,22,48], where the equality constraint is already implied in each attribute. Although we use the same names, the concepts of support and confidence for matching dependencies have no relation to the support and confidence for association rules [2]. The measures for association rules are defined on the number of occurrences of item sets in transactions, while the measures for data dependencies [11] consider the number of tuple pairs, in particular, with respect to similarity metrics in MDs. Indeed, association rules specify constraints at instance level (item) with respect to one record (transaction). Matching dependencies [16] consider constraints between two records (tuples) where no instance values are specified.

1.2. Contributions

In this paper, given a relation instance and $X \rightarrow Y$, we study the issues of discovering similarity constraints for matching dependencies on the given $X \rightarrow Y$. Our main contributions are summarized as follows:

- First, we introduce the utility evaluation for MDs. Specifically, the confidence and support measures of matching dependencies are formally defined, together with the computation of measures in the given relation instance.
- Second, we study exact algorithms for discovering MDs. The MDs discovery problem is to find settings of similarity thresholds on attributes for MDs that can satisfy the required confidence and support. We first present an exact solution and then study pruning strategies by the minimum requirements of support and confidence.
- Third, we study approximation algorithms for discovering MDs. Since the exact algorithm has to traverse all the data during the computation, we propose an approximate solution which only uses some of the data. A bound of relative errors introduced by the approximation is developed. Moreover, we also develop a strategy of early termination.
- Finally, we report an extensive experimental evaluation. The proposed algorithms on discovering MDs are studied. The experimental results demonstrate that our pruning strategies can significantly improve the discovery efficiency.

The remainder of this paper is organized as follows. First, Section 3 presents the utility measures for MDs, including support and confidence. In Section 4, we develop the exact algorithm for discovering MDs and study the corresponding pruning strategies. In Section 5, we present the approximation algorithm with bounded relative errors. Then, Section 6 reports our extensive experimental evaluation. Finally, we introduce some related work in Section 2, and conclude this paper in Section 7. Table 2 lists the frequently used notations in this paper. An earlier and shorter conference version of this paper appears in [43].

2. Related work

Recently, traditional data dependencies, such as functional dependencies (FDs) and inclusion dependencies (INDs) for the schema design [1], are revisited for new applications like improving the quality of data [16] or privacy-preserving [47]. The

Table 2
Notations.

Symbol	Description
φ	Matching dependency, MD
λ	Threshold pattern, of similarity
$\mathcal{C}_t$	Candidate set of threshold patterns
c	$\lvert\mathcal{C}_t\rvert$, number of threshold patterns in $\mathcal{C}_t$
η_s	Minimum requirement, of support
η_c	Minimum requirement, of confidence
$\mathcal{R}$	Original relation of data tuples
t	Original data tuple
N	Number of data tuples in original relation
$\mathcal{D}$	Statisical distribution of statistical tuples
s	Statistical tuples for matching quality
n	Number of statistical tuples in $\mathcal{D}$

conditional functional dependencies (CFDs), as an extension of traditional FDs with conditions, are first proposed in [8] for data cleaning. The basic idea of CFDs is making the FDs, originally hold for the whole table, valid only for a set of tuples specified by the conditions. Cong et al. [14] study the detecting and repairing methods of violations to CFDs. Fan et al. [21] investigate the propagation of CFDs for data integration. Bravo et al. [9] propose an extension of CFDs by employing disjunction and negation. Golab et al. [26] define a range tableau for CFDs, where each value is a range. In addition, Bravo et al. [10] propose conditional inclusion dependency (CINDs), which are useful not only in data cleaning, but are also in contextual schema matching. Ilyas et al. [30] study a novel soft FD, which is also a generalization of the classical notion of a hard FD where the value of X completely determines the value of Y. In a soft FD, the value of X determines the value of Y not with certainty, but merely with high probability. All the above extensions of dependencies are still based on the equality function, while our work studies the matching dependencies by incorporating similarity metrics.

2.1. Dependencies with metrics

Matching dependencies (MDs) are first proposed in [16] for specifying matching rules for the object identification (a.k.a. record matching or entity resolution, see [15] for a survey). The MDs can be regarded as a generalization of FDs, which are based on the equality of values (i.e., having matching similarity equal to 1.0 exactly). Thus, FDs can be represented by the syntax of MDs as well. Reasoning mechanism for deducing MDs from a set of given MDs is studied in [19]. A sound and complete inference system is also presented in [17] for the deduction analysis of MDs. Matching dependencies and matching keys together with the reasoning techniques can improve the quality of record matching methods. ul Hassan et al. [46] utilize matching dependencies for improving the quality of consolidation in linked dataspaces. Data repairing and consistent query answering with matching dependencies are also studied [23,5]. Moreover, Fan et al. [20] study the repairing with the cooperation of both conditional functional dependencies and matching dependencies. Motivated by the usefulness of MDs, in this paper, we study the problem of discovering similarity constraints for matching dependencies from data.

Besides matching dependencies, in recent work, the importance of introducing similarity metrics in dependencies has been commonly recognized. Koudas et al. [33] study the dependencies with similarity metrics on attributes Y when given the equal values on X. In addition, Song and Chen [44] apply distance/similarity metrics on both sides of X and Y attributes, known as differential dependencies. Bassée and Wijsen [3] propose neighborhood dependencies (NDs) for predication purpose. Intuitively, NDs express the constraint that if two tuples are close with respect to the predictor variables, then these two tuples should have similar values for the target variable. Their extensions of dependencies with similarity metrics have been found useful. Unfortunately, the discovery of similarity thresholds in such dependencies is not studied in previous work.

In addition to the metric distance/similarity of values, Golab et al. [25] propose sequential dependencies, which targets on ordered data. A sequential dependency, in the form of $X \rightarrow_g Y$, states that when tuples are sorted on X, the distance between the Y-values of any two consecutive tuples are within interval g. The matching dependencies considered in our work do not require the data that can be ordered.

2.2. Discovery of dependencies

The discovery of dependencies from a given relation instance is widely studied [7,36,37,42,34,31]. In discovering FDs, previous work targeted on generating a canonical cover of all FDs.

Due to the inherent hardness of the discovery problem, i.e., the size of results could be exponential [1], a series of strategies have been proposed to improve the efficiency of discovery. Huhtala et al. [28,29] propose a level-wise algorithm, namely TANE, together with efficient pruning techniques for searching in the lattice of attributes. Remarkably, TANE algorithm also supports the discovery of approximate FDs, where the efficient pruning technique can still be applied. Instead of the level-wise searching, Wyss et al. [48] study depth-first, heuristic-driven algorithm, namely FastFDs, which is (almost) linear to the size of FDs cover. Rather than pruning the valid non-minimal dependencies, Flach and Savnik [22] discover FDs in a bottom-up style, which considers the maximal invalid dependencies first. When searching in a hypotheses space, the maximum invalid dependencies are used for pruning the search space. The above three strategies are also adapted to improve the efficiency of discovering CFDs, by Fan et al. [18].

Unfortunately, the problem of discovering similarity thresholds for MDs studied in this paper, is not considered in the existing techniques on discovering FDs. Once the attributes X and Y in the dependencies are determined. It already implies the equality constraint on each attribute. Therefore, the existing techniques are not applicable to improve the efficiency of discovering the similarity thresholds for MDs.

2.3. Measures for dependencies

A dependency can be measured in various ways. In measures of approximate FDs, since a FD may "almost" hold on a relation instance, g_3 measure [32] is widely used to evaluate these approximate FDs [28,29,26], that is, the minimum number of tuples that have to be removed from the relation instance for the FD to hold. Pfahringer and Kramer [40,34] propose a measure based on the minimum description length principal. An encoding length of a table T is defined based on $X \rightarrow Y$ to compress T. Giannella and Robertson [24] develop an approximation measure of FDs based on the intuition: the degree to which $X \rightarrow Y$ is approximate in a table T is the degree to which T determines a function from $\prod_X(T)$ to $\prod_Y(T)$. In addition, Chiang and Miller [12] also study some other measures such as conviction and χ^2-test for evaluating dependency rules. Most of the previous measures are defined on the equality

function for FDs, which are not directly applicable to MDs with similarity metrics. For example, the g_3 measure is computed by grouping tuples with equal values, while the similarity metrics are associated with respect to tuple pairs without groups.

The confidence and support measures, which are defined based on tuple pairs, are also used in discovering approximate functional dependencies [11]. The confidence can be interpreted as an estimate of the probability that a randomly drawn pair of tuples agreeing on X also agree on Y. As mentioned in the Introduction, the support and confidence measures [41] for finding association rules [2] are different from the measures used for evaluating data dependencies [11]. In short, the measures for association rules are defined on the number of occurrences of item sets in transactions, while the measures for data dependencies consider the number of tuple pairs with respect to distance metrics. Following [11], in our study, we also utilize support and confidence to evaluate MDs.

3. Utility measures

In this section, we formally introduce the definitions of MDs. Then, we investigate utility measures for evaluating MDs over a given database instance.

3.1. Matching dependencies

Traditional functional dependencies (FDs and their extensions) rely on the equality operator $=$ to identify dependency relationships. However, in some real world application, it is not possible to use the equality operator $=$ to identify matching over fuzzy data values such as text values. For instance, Jason Smith and J.Smith in the Name attribute may refer to the same real world entity. Therefore, *matching dependencies* (MDs) [16] are proposed based on the matching quality. Instead of the equality constraint in FDs, the MDs declare similarity constraints through the similarity matching operators, denoted by $\approx$, such as *edit distance* [39], *cosine similarity* with word tokens [13] or *q-grams* [27] for text values.

Consider a relation r with schema $\mathcal{R}(A_1, \ldots, A_m)$. Following similar syntax of FDs, we define MDs as following.[1] A *matching dependency* (MD) φ has the form $(X \rightarrow Y, \lambda)$, where $X\ \mathcal{R}, Y\ \mathcal{R}$ are two sets of attributes, and λ is a *threshold pattern* of similarity thresholds on attributes in $X \cup Y$, e.g., $\lambda[A]$ denotes the similarity threshold on attribute $A \in X \cup Y$.

Definition 1. A Matching Dependency (MD) φ specifies a constraint that, for any two tuples t_1 and t_2 in a relation r with schema $\mathcal{R}$, if $\Lambda_{A_i \in X} t_1[A_i] \approx_{\lambda[A_i]} t_2[A_i]$, then $\Lambda_{A_j \in Y} t_1[A_j] \approx_{\lambda[A_j]} t_2[A_j]$, where $\lambda[A_i]$ and $\lambda[A_j]$ are the similarity thresholds on the attributes of A_i and A_j respectively. For each attribute $A_i \in X \cup Y$, the similarity matching operator $\approx$ returns true, if the similarity between $t_1[A_i]$ and $t_2[A_i]$ satisfies the corresponding threshold $\lambda[A_i]$.

For example, an MD φ([Street] → [City], <0.8, 0.7>) in the Contacts relation denotes that if two tuples have similar Street (with matching similarity greater than 0.8) then their City values are probably similar as well (with similarity at least 0.7).

3.2. Measures

Following [11], we adopt the *support* and *confidence* measures to evaluate the matching dependencies over a relation instance. It is worth noting that the measures for FDs, e.g., g_3 measure [32], can be directly computed on the data tuples, based on the equality of tuple values. However, for the above similarity constraints of MDs, we need to consider the matching quality (e.g., cosine similarity or edit distance) of all the pairs of tuples t_1 and t_2 for $\mathcal{R}$. When evaluating the measures of different MDs, instead of on-line computing the matching quality, we can pre-compute the pair-wised tuple matching off-line, and store the results for reuse among MDs.

Therefore, we introduce a statistical distribution (denoted by $\mathcal{D}$) for the matching quality of pair-wised tuple matching for $\mathcal{R}$. The statistical distribution has a schema $\mathcal{D}(A_1, \ldots, A_m, P)$, where each attribute A_i in $\mathcal{D}$ records similarities between values on the attribute A_i in $\mathcal{R}$, and P is the statistical value. Let s be a statistical tuple in $\mathcal{D}$, and $s[A_i]$ be the projection on attribute A_i of s. Each $s[A_i]$ denotes a similarity value of two tuples on attribute A_i in $\mathcal{R}$. The statistic $s[P]$ denotes the probability that any two tuples t_1 and t_2 of $\mathcal{R}$ have the similarity values $s[A_i]$, $\forall A_i \in \mathcal{R}$. With a pair-wised evaluation of matching quality of all the N tuples for $\mathcal{R}$, we can easily compute P by $\frac{count(s)}{N*(N-1)/2}$, where $count(s)$ denotes the number of pairs of tuples having the similarity values in s. Different similarity matching operators have various spaces of similarity values, such as cosine similarity in [0.0, 1.0] while edit distance having edit operations 1,2,.... In order to evaluate in a consistent environment, we map these similarity values $s[A]$ to a unified space, say $[0, d-1]$, which is represented by $\text{sim}(A)$ with d elements.

Example 1. Table 3 shows an example of the statistical distribution $\mathcal{D}$ computed from Contacts in Table 1. First, cosine similarities in the range of [0.0, 1.0] are mapped to elements in $[0, d-1]$ of $\text{sim}(A)$ with $d = 11$, i.e., the cosine similarity value times $d-1$. For example, the cosine similarity between t_1[Name] and t_2[Name] in Table 1 is 0.7. This similarity value 0.7 in the range of [0.0, 1.0] is mapped to a unified space [0, 10], i.e., $0.7 * (d-1) = 7$. By computing the similarities on all the attributes between t_1 and t_2, we can obtain a similarity vector <10, 7, 10, 10, 10, 10>. It states that the similarities on attributes SIN, Name,..., Street between t_1 and t_2 are 10, 7, ..., 10, respectively. Considering all the $6 * 5 = 30$ distinct tuple pairs in Table 1, there is no other tuple pair

[1] The MDs syntax is described with two relation schema R_1, R_2 for object identification in [16], which can also be represented in a single relation schema R as the FDs.

Table 3
Example of statistical distribution $\mathcal{D}$.

A_1	A_2	A_3	A_4	A_5	A_6	P	
10	7	10	10	10	10	0.033	s_1
10	8	10	10	10	8	0.033	s_2
0	0	0	0	0	8	0.066	s_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

having the same similarity vector. Consequently, the first tuple $s_1(10, 7,..., 10, 0.033)$ in Table 3 denotes that there are about 3.3%, i.e., $s_1[P] = \frac{1}{30}$, matching pairs in all pair-wised tuple matching, whose similarity values are 10, 7,…,10 on attributes SIN, Name,…, Street (corresponding to A_1, A_2,..., A_6), respectively.

Similarly, for the tuple pair (t_1,t_3), we compute a similarity vector <10, 8, 10, 10, 10, 8>, which yields a statistical tuple $s_2(10, 8,..., 8, 0.033)$. The next tuple $s_3(0, 0,..., 8, 0.066)$ is generated by the tuple pairs (t_1,t_4) and (t_2,t_4) with the same similarity vector <0, 0, 0, 0, 0, 8>.

Then, we can measure the support and confidence of MDs, with various attributes X and Y, based on the statistical distribution $\mathcal{D}$. Let λ_X and λ_Y be the projections of similarity threshold pattern λ on the attributes of X and Y, respectively, in an MD φ. It is worth noting that the similarity thresholds are also specified in terms of elements in $\mathrm{sim}(A)$ of each $A \in X \cup Y$. Let Z be the set of attributes not specified by φ, i.e., $\mathcal{R} \setminus (X \cup Y)$. The definitions of support and confidence for the MD $\varphi(X \rightarrow Y, \lambda)$ are presented as follows:

$$\mathrm{support}(\varphi) = P(X \vDash \lambda_X, Y \vDash \lambda_Y) = \sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z) \tag{1}$$

$$\mathrm{confidence}(\varphi) = P(Y \vDash \lambda_Y | X \vDash \lambda_X) = \frac{\sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z)}{\sum_{Y,Z} P(X \vDash \lambda_X, Y, Z)} \tag{2}$$

where $\vDash$ denotes the *satisfiability* relationship, i.e., $X \vDash \lambda_X$ denotes that the similarity values on all attributes in X satisfy the corresponding thresholds listed in λ_X. For example, we say that a statistical tuple s in $\mathcal{D}$ satisfies λ_X, i.e., $s[X] \vDash \lambda_X$, if s has similarity values higher than the corresponding minimum threshold, i.e., $s[A] \geq \lambda[A]$, for each attribute A in X. Consequently, we can calculate $\sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z)$ in formula (1) by $\sum_{s_i \in \mathcal{D}, s_i[X] \vDash \lambda_X, s_i[Y] \vDash \lambda_Y} s_i[P]$, i.e., the proportion of tuple pairs whose similarities on attributes X and Y satisfy the corresponding thresholds λ_X and λ_Y. Similarly, $\sum_{Y,Z} P(X \vDash \lambda_X, Y, Z)$ in formula (2) can be computed by $\sum_{s_i \in \mathcal{D}, s_i[X] \vDash \lambda_X} s_i[P]$.

Consider any two tuples t_1 and t_2 from the original data relation $\mathcal{R}$, the support(φ) estimates the probability that the matching similarities of t_1 and t_2 on attributes X and Y satisfy the thresholds specified by λ_X and λ_Y, respectively. Similarly, the confidence(φ) computes the conditional probability that the matching similarities between t_1 and t_2 on Y satisfy the thresholds specified by λ_Y (i.e., $Y \vDash \lambda_Y$) given the condition that t_1 and t_2 are similar on attributes X (i.e., $X \vDash \lambda_X$). Thus, high confidence(φ) means few instances of matching pairs that are similar on attributes X (i.e., $X \vDash \lambda_X$) but not similar on attributes Y (i.e., $Y \nvDash \lambda_Y$), where $\nvDash$ denotes the unsatisfiability relationship.

In real applications inconsistency detection, in order to achieve high detection accuracy, we would like to use MDs with high confidence. On the other hand, if users need high recall of detection, then MDs with high support are preferred. Intuitively, we would like to discover those MDs with high support and high confidence. Therefore, in the rest of this paper, we study the problem of discovering similarity constraints for MDs that can satisfy users' minimum utility requirements of support η_s and confidence η_c.

4. Exact algorithms

We now study the determination of similarity threshold pattern for MDs based on the statistical distribution. Recognize that the problem of determining similarity constraints is new and different from the previous FDs discovery [28,22,48]. Indeed, once the $X \rightarrow Y$ is given for a FD, it already implies the similarity threshold to be 1.0, that is, $(X \rightarrow Y, <1.0, 1.0>)$ using the MD syntax. Unlike FDs, we have various settings of matching similarity thresholds for MDs. Therefore, given the statistical distribution processed from a relation, we discover the proper similarity thresholds for MDs such that the required support and confidence are satisfied.

4.1. Problem statement

In order to discover an MD φ with the minimum requirements of support η_s and confidence η_c, the following preliminary should be given first: (I) what is Y? and (II) what is matching quality requirement λ_Y. These two preliminary questions are usually addressed by specific applications. For example, if we would like to use the discovered MDs to guide object identification in the Contacts table, then $Y =$ SIN. The λ_Y is often set to high similarity thresholds by applications to ensure high matching quality on Y attributes. For example, λ_Y is set to 1.0 for $Y =$ SIN in the object identification application. Note that without the preliminary λ_Y, the discovered MDs will be meaningless. For example, an MD with $\lambda_Y = 0$ can always satisfy any requirement of η_c. Since all the

statistical tuples can satisfy the thresholds $\lambda_Y = 0$, the corresponding confidence will always be equal to 1.0. Obviously, such MD with $\lambda_Y = 0$ is useless and cannot detect any violation or identify objects correctly.

Formally, consider a statistical distribution $\mathcal{D}$. The threshold determination problem of MDs is:

Problem 1. Given the attributes X and Y, the minimum requirements of support and confidence η_s, η_c, and the similarity threshold pattern λ_Y, to find all the MDs $\varphi(X \to Y, \lambda)$ with threshold pattern λ_X on attributes X having confidence$(\varphi) \geq \eta_c$ and support$(\varphi) \geq \eta_s$, if exist; otherwise return infeasible.

The attributes X can be initially assigned to $\mathcal{R} \setminus Y$ if no suggestion is provided by specific applications, since our discovery process can automatically remove those attributes that are not required in X for an MD φ. Specifically, when a possible discovered threshold $\lambda[A]$ on attribute A is $0 \in \text{sim}(A)$, it means that any similarity value of the attribute $A \in X$ can satisfy the threshold 0 and will not affect the MD φ at all. In other words, the attribute A can be removed from X in the MD φ, if $\lambda[A]$ is 0.

4.2. Exact algorithm

We now present an algorithm to compute the similarity thresholds on attributes X for MDs having support and confidence greater than η_s and η_c, respectively. Let $A_1, \ldots, A_{m_X}$ be the m_X attributes in X, which can be simply assigned to $\mathcal{R} \setminus Y$ as aforesaid. For simplicity, we use λ to denote the threshold pattern projection λ_X with $\lambda[A_1], \ldots, \lambda[A_{m_X}]$ on all the m_X attributes of X. Since we only use the values from $\text{sim}(A_i)$ as possible thresholds $\lambda[A_i]$ on attribute A_i, i.e., $\lambda[A_i] \in \text{sim}(A_i)$, we can investigate all the possible candidates of threshold pattern λ. Let $\mathcal{C}_t$ be the set of all the possible threshold pattern candidates, having

$$\mathcal{C}_t = \text{sim}(A_1) \times \cdots \times \text{sim}\left(A_{m_X}\right) = \text{sim}(X).$$

Given a fixed scheme $\mathcal{R}$, the total number of candidates is $c = |\mathcal{C}_t| = |\text{sim}(X)| = d^m$, where d is the size of $\text{sim}(A_i)$.

Let n be the number of statistical tuples in the input statistical distribution $\mathcal{D}$. We consider two statistical values $P_i^j(X, Y)$ and $P_i^j(X)$, which record $P(X \vDash \lambda_X, Y \vDash \lambda_Y)$ and $P(X \vDash \lambda_X)$ respectively for the candidate $\lambda_j \in \mathcal{C}_t$ based on the information of the first i tuples in $\mathcal{D}$, initially having $P_0^j(X, Y) = P_0^j(X) = 0$. The recursion is defined as follows, with i increasing from 1 to n and j increasing from 1 to c.

$$P_i^j(X, Y) = \begin{cases} P_{i-1}^j(X, Y) + s_i[P], & \text{if } s_i[X] \vDash \lambda_j, s_i[Y] \vDash \lambda_Y \\ P_{i-1}^j(X, Y), & \text{otherwise} \end{cases}$$

$$P_i^j(X) = \begin{cases} P_{i-1}^j(X) + s_i[P], & \text{if } s_i[X] \vDash \lambda_j \\ P_{i-1}^j(X), & \text{otherwise} \end{cases}.$$

Finally, those λ_j can be returned if support$(\lambda_j) = P_n^j(X, Y) \geq \eta_s$ and confidence$(\lambda_j) = \frac{P_n^j(X, Y)}{P_n^j(X)} \geq \eta_c$.

Algorithm 1. Exact Algorithm

EA $(\mathcal{D}, \mathcal{C}_t, \lambda_Y)$

Algorithm 1 Exact Algorithm
EA$(\mathcal{D}, C_t, \lambda_Y)$

Input: Statistical distribution $\mathcal{D}$, Candidate set C_t, and λ_Y
Output: All the valid λ_j

```
1: for each candidate λ_j ∈ C_t, j : 1 → c do
2:     P_0^j(X, Y) = P_0^j(X) = 0
3:     for each statistical tuples s_i ∈ D, i : 1 → n do
4:         compute P_i^j(X) and P_i^j(X, Y) for λ_j, λ_Y
5:     end for
6: end for
7: return all the λ_j with confidence and support satisfying η_c, η_s
```

We can implement the exact algorithm (namely EA) by considering all the statistical tuples s_i in $\mathcal{D}$ with i from 1 to n, with the complexity $\mathcal{O}(nc)$.

4.3. Pruning strategies

The original exact algorithm needs to traverse all the n statistical tuples in $\mathcal{D}$ and c candidate threshold patterns in $\mathcal{C}_t$, which is very costly. Indeed, with the given η_s and η_c, we can investigate the relationship between similarity thresholds and avoid checking

all candidate threshold patterns in $\mathcal{C}_t$ and all statistical tuples in $\mathcal{D}$. Therefore, in the following, we present two pruning techniques based on the given support and confidence, respectively.

4.3.1. Pruning by support

We first study the relationships among different threshold patterns, based on which we then propose rules to filter out candidates that have supports lower than η_s.

Definition 2. Given two similarity threshold patterns λ_1 and λ_2, if $\lambda_1[A] \leq \lambda_2[A]$ holds for all the attributes, $\forall\, A \in X$, then λ_1 dominates λ_2, denoted as $\lambda_1 \preccurlyeq \lambda_2$.

Based on the dominant definition, the following Lemma describes the relationships of supports between similarity threshold patterns.

Lemma 1. *Given two* MDs, $\varphi_1 = (X \rightarrow Y, \lambda_1)$ *and* $\varphi_2 = (X \rightarrow Y, \lambda_2)$ *over the same relation instance of* $\mathcal{R}$, *if* λ_1 *dominates* λ_2, $\lambda_1 \preccurlyeq \lambda_2$, *then we have*

$$\text{support}(\varphi_1) \geq \text{support}(\varphi_2).$$

Proof. Let $\text{cover}(\lambda_1)$ and $\text{cover}(\lambda_2)$ denote the set of statistical tuples that satisfy the threshold λ_1 and λ_2 respectively, e.g.,

$$\text{cover}(\lambda_2) = \{s | s[X] \models \lambda_2, s \in \mathcal{D}\}.$$

According to the minimum similarity thresholds, for each attribute A, we have $\lambda_2[A] \leq s[A]$. In addition, since $\lambda_1 \preccurlyeq \lambda_2$, for any tuple $s \in \text{cover}(\lambda_2)$, we also have $\lambda_1[A] \leq \lambda_2[A] \leq s[A]$ on all the attributes A. In other words, the set of statistical tuples covered by λ_2 also satisfy the threshold of λ_1, i.e.,

$$\text{cover}(\lambda_2) \subseteq \text{cover}(\lambda_1).$$

Referring to the definition of support, we have

$$\text{support}(\varphi_1) \geq \text{support}(\varphi_2).$$

The conclusion is proved. □

According to Lemma 1, given a candidate similarity threshold pattern λ_j having lower support than the user specified requirement η_s, i.e., $P_n^j(X, Y) < \eta_s$, all the candidates that are dominated by λ_j should also have support lower than η_s and can be safely pruned without computing their associated support and confidence.

We present the implementation of pruning by support (namely EPS) in Algorithm 2. As illustrated, the approach extends Algorithm 1 by introducing the pruning step in Lines 6–8. If the current λ_j is not valid, then all the remaining candidate λ' that are dominated by λ_j can be safely pruned in Line 7.

In order to maximize the pruning, we can heuristically select an ordering of candidates in $\mathcal{C}_t$ that for any $j_1 < j_2$ having $\lambda_{j_1} \preccurlyeq \lambda_{j_2}$. That is, we always first process the candidates that dominate others. In fact, we can use a directed acyclic graph to represent candidate similarity patterns as vertices and dominant relationships among the similarity patterns as edges. Therefore, the dominant order of candidate patterns can be obtained by a BFS traversal upon the graph.

Algorithm 2. Pruning by Support

EPS $(\mathcal{D}, \mathcal{C}_t, \lambda_Y)$

Algorithm 2 Pruning by Support
EPS$(\mathcal{D}, C_t, \lambda_Y)$

Input: Statistical distribution $\mathcal{D}$, Candidate set C_t, and λ_Y
Output: All the valid λ_j

1: **for** each candidate $\lambda_j \in C_t, j : 1 \rightarrow c$ **do**
2: $P_0^j(X, Y) = P_0^j(X) = 0$
3: **for** each tuple $s_i \in \mathcal{D}, i : 1 \rightarrow n$ **do**
4: compute $P_i^j(X)$ and $P_i^j(X, Y)$ for λ_j, λ_Y
5: **end for**
6: **if** $P_n^j(X, Y) < \eta_s$ **then**
7: remove all the remaining candidates λ' dominated by λ_j from C_t {Pruning by support, $\lambda' \succcurlyeq \lambda_j$}
8: **end if**
9: **end for**
10: **return** all the λ_j with confidence and support satisfying η_c, η_s

4.3.2. Pruning by confidence

Other than pruning by support, we can also utilize the given confidence requirement to avoid further examining the statistical tuples that have no improvement of confidence, when the confidence is already lower than η_c for a candidate λ_j.

First, we divide the statistical tuples in $\mathcal{D}$ into two parts based on the preliminary λ_Y as follows. Let v be a pivot between 1 and n. For the first v tuples, we have

$$s_i[Y] \models \lambda_Y, 1 \le i \le v,$$

while all the remaining n–v tuples have

$$s_i[Y] \not\models \lambda_Y, v+1 \le i \le n.$$

This partitioning of statistical tuples in $\mathcal{D}$ can be done in linear time.

Lemma 2. *Consider a pre-partitioned statistical distribution $\mathcal{D}$. For any $1 \le i_1 < i_2 \le n$, we always have*

$$\frac{P^j_{i_1}(X,Y)}{P^j_{i_1}(X)} \ge \frac{P^j_{i_2}(X,Y)}{P^j_{i_2}(X)}.$$

Proof. Since the first v tuples have $s_i[Y] \models \lambda_Y$, according to the computation of $P(X, Y)$ and $P(X)$, we have

$$\frac{P^j_i(X,Y)}{P^j_i(X)} = 1.0, \quad 1 \le i \le v.$$

Moreover, for the remaining n–v tuples with $s_i[Y] \not\models \lambda_Y$, the $P(X, Y)$ value will not change any more, i.e.,

$$P^j_i(X,Y) = P^j_v(X,Y), v+1 \le i \le n.$$

Meanwhile, the corresponding $P(X)$ is non-decreasing, that is,

$$P^j_v(X) \le P^j_{i_1}(X) \le P^j_{i_2}(X)$$

for any $v + 1 \le i_1 < i_2 \le n$. Consequently, we have

$$\frac{P^j_{i_1}(X,Y)}{P^j_{i_1}(X)} \ge \frac{P^j_{i_2}(X,Y)}{P^j_{i_2}(X)}, \quad v+1 \le i_1 < i_2 \le n.$$

Combining above two statements, we proved the lemma. □

Therefore, according to the formula of confidence, with the increase of i from 1 to n, the confidence of a specific candidate λ_j is non-increasing. For a candidate λ_j, when processing the statistical tuple s_i, if the current confidence $\frac{P^j_i(X,Y)}{P^j_i(X)}$ is lower than η_c, then we can prune the candidate λ_j without considering the remaining statistical tuples from $i + 1$ to n in $\mathcal{D}$.

Finally, both the pruning by support and the pruning by confidence are cooperated together into a single threshold determination algorithm as shown in Algorithm 3 (namely EPSC). We demonstrate the performance of the hybrid pruning EPSC in Section 6.

Algorithm 3. Pruning by Support & Confidence

EPSC $(\mathcal{D}, \mathcal{C}_t, \lambda_Y)$

Algorithm 3 Pruning by Support & Confidence
EPSC$(\mathcal{D}, C_t, \lambda_Y)$

Input: Statistical distribution $\mathcal{D}$, Candidate set C_t, and λ_Y
Output: All the valid λ_j

1: **for** each candidate $\lambda_j \in C_t$, $j : 1 \to c$ **do**
2: $P^j_0(X,Y) = P^j_0(X) = 0$
3: **for** each tuple $s_i \in \mathcal{D}, i : 1 \to n$ **do**
4: compute $P^j_i(X)$ and $P^j_i(X,Y)$ for λ_j, λ_Y
5: **if** $\frac{P^j_i(X,Y)}{P^j_i(X)} < \eta_c$ **then**
6: remove λ_j from C_t {Pruning by confidence}
7: **if** $P^j_i(X,Y) \ge \eta_s$ **then**
8: **break**
9: **end if**
10: **end if**
11: **end for**
12: **if** $P^j_n(X,Y) < \eta_s$ **then**
13: remove all the remaining candidates λ' dominated by λ_j from C_t {Pruning by support, $\lambda' \gg \lambda_j$}
14: **end if**
15: **end for**
16: **return** all the λ_j with confidence and support satisfying η_c, η_s

5. Approximation algorithms

Though we have proposed the pruning of candidate patterns for the exact method (Algorithm 3), the evaluation space of each candidate is still all the n statistical tuples in statistical distribution $\mathcal{D}$. Therefore, in this section, we present an approximation algorithm which only traverses the first k ($k = 1, \ldots n$) tuples in $\mathcal{D}$, with bounded relative errors on support and confidence of returned MDs.

5.1. Preliminary

Let C^n and S^n be the confidence and support computed in the exact solution with all n statistical tuples. We study the approximate confidence and support, C^k and S^k, by ignoring the statistical tuples from s_{k+1} to s_n. The decision of k will be discussed soon below.

For any candidate threshold pattern $\lambda_j \in \mathcal{C}_t$, let

$$\beta = P_k^j(X),$$

$$\overline{\beta} = P_n^j(X) - P_k^j(X),$$

where β denotes $P(X \vDash \lambda_X)$ for the candidate λ_j based on the first k tuples in $\mathcal{D}$, and $\overline{\beta}$ is $P(X \vDash \lambda_X)$ based on the remaining n–k tuples. The following Lemma indicates the error bounds of C^k and S^k when $\overline{\beta}$ for a specific k is in a certain range.

Lemma 3. *If we have*

$$\overline{\beta} \le \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right),$$

then the error of approximate confidence C^k compared to the exact confidence C^n is bounded by

$$-\epsilon \le \frac{C^n - C^k}{C^n} \le \epsilon,$$

and the error of approximate support S^k compared to the exact S^n is bounded by

$$\frac{S^n - S^k}{S^n} \le \epsilon.$$

Proof. Let

$$\alpha = P_k^j(X, Y),$$
$$\overline{\alpha} = P_n^j(X, Y) - P_k^j(X, Y).$$

According to the computation of confidence, we have $C^k = \frac{\alpha}{\beta}$ and $C^n = \frac{\alpha+\overline{\alpha}}{\beta+\overline{\beta}}$. Let $Z = 1 - \frac{C^n - C^k}{C^n} = \frac{C^k}{C^n}$, that is,

$$Z = \frac{\alpha(\beta+\overline{\beta})}{\beta(\alpha+\overline{\alpha})} \le 1 + \frac{\overline{\beta}}{\beta}.$$

First, we have

$$\beta = \alpha + \sum_{i=1}^{k} s_i \left[P\left(X \vDash \lambda_j, Y \nvDash \lambda_Y\right)\right] \ge \alpha.$$

Note that α is equal to the approximate support S^k of the MD φ with similarity threshold pattern λ_j on the attributes X. According to the minimum support requirement, to discover a valid λ_j, we need the approximately computed support S^k to be greater than the minimum support η_s, i.e., $\alpha = S^k \ge \eta_s$. Consequently, it follows $\beta \ge \alpha \ge \eta_s$. Thereby, we have

$$Z \le 1 + \frac{\overline{\beta}}{\eta_s}.$$

Moreover, according to the condition $\overline{\beta} \le min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$, that is $\overline{\beta} \le \epsilon\eta_s$, we have

$$Z \le 1 + \epsilon.$$

Second, similar to $\beta \geq \alpha$, we also have $\overline{\alpha} \leq \overline{\beta}$ for the statistical tuples from $k+1$ to n. Therefore,

$$Z \geq \frac{\alpha(\beta+\overline{\beta})}{\beta(\alpha+\overline{\beta})} = \frac{\beta+\overline{\beta}}{\beta+\frac{\beta\overline{\beta}}{\alpha}}.$$

According to the minimum confidence $\frac{\alpha}{\beta} \geq \eta_c$,

$$Z \geq \frac{\beta+\overline{\beta}}{\beta+\frac{\overline{\beta}}{\eta_c}} = 1-\frac{\overline{\beta}(1-\eta_c)}{\beta\eta_c+\overline{\beta}}. \tag{3}$$

Recall that $\beta \geq \eta_s$ and the confidence should be lower than or equal to 1, i.e., $\eta_c \leq 1$. Thus,

$$Z \geq 1-\frac{\overline{\beta}(1-\eta_c)}{\eta_s\eta_c+\overline{\beta}} = 1-\frac{1-\eta_c}{\frac{\eta_c\eta_s}{\overline{\beta}}+1}.$$

Since we have the condition, it follows $\overline{\beta} \leq \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}$,

$$Z \geq 1-\frac{1-\eta_c}{\frac{1-\epsilon-\eta_c}{\epsilon}+1} = 1-\epsilon.$$

Finally, based on the above two conditions, we conclude that

$$1+\epsilon \geq Z = 1-\frac{C^n-C^k}{C^n} = \frac{C^k}{C^n} \geq 1-\epsilon,$$
$$-\epsilon \leq \frac{C^n-C^k}{C^n} \leq \epsilon.$$

On the other hand, according to the computation of support, we have $S^k = \alpha$ and $S^n = \alpha+\overline{\alpha}$. Therefore,

$$\frac{S^n-S^k}{S^n} = \frac{1}{1+\frac{\alpha}{\overline{\alpha}}}.$$

Recall that we have $\alpha \geq \eta_s$ and $\overline{\alpha} \leq \overline{\beta} \leq \epsilon\eta_s$. It can be rewritten by

$$\frac{S^n-S^k}{S^n} \leq \frac{1}{1+\frac{1}{\epsilon}} = \frac{\epsilon}{1+\epsilon} < \epsilon.$$

To sum up, the worst-case relative error is bounded by ϵ for both the confidence and support. □

5.2. Approximation algorithm

Now, we consider the last n–k tuples in $\mathcal{D}$. Let

$$\overline{B}(k) = \sum_{i=k+1}^{n} s_i[P],$$

where $s_i[P]$ is the probability associated to each statistical tuple in $\mathcal{D}$. Referring to the definition of $\overline{\beta}$, for any λ_j, we always have $\overline{\beta} \leq \overline{B}(k)$. If there exists a k having $\overline{B}(k) \leq \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$, then $\overline{\beta} \leq \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$ is satisfied for all the threshold candidates λ_j. Since the $\overline{B}(k)$ decreases with the increase of k, to determine a minimum k is to find a corresponding maximum $\overline{B}(k)$. Therefore, according to Lemma 3, given an error bound ϵ, $0 < \epsilon < 1-\eta_c$, we can compute a minimum position

$$k = \arg\max_{k=1}^{n} \overline{B}(k), \tag{4}$$

that is, the k in the range of $[1,n]$ such that $\overline{B}(k)$ attains the maximum value, having $\overline{B}(k) \leq \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$.

Theorem 1. *Given an error bound ϵ, $0 < \epsilon < 1 - \eta_c$, we can determine a minimum k, having*

$$\overline{B}(k) \le \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right), 1 \le k \le n.$$

The approximation by considering the first k statistical tuples in $\mathcal{D}$ finds approximate MDS *with the error bound ϵ on both the confidence and support compared with the exact one. The complexity is $\mathcal{O}(kc)$.*

We present the approximation implementation in Algorithm 4. Let $\overline{B}$ denote $\overline{B}(k) = \sum_{i=k+1}^{n} s_i[P]$ for the current k. With k decreasing from n to 1, we can determine a minimum k where $\overline{B} = \overline{B}(k) \le \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$ is still satisfied. After computing k, we process the tuples s_i starting from $i = 1$. When the bound condition is first satisfied, i.e., $i = k$ with $\overline{B} = \overline{B}(k) \le \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$, the processing terminates. Here, the error bound ϵ is specified by user requirement with $0 < \epsilon < 1 - \eta_c$.

Note that no particular ordering of statistical tuples in $\mathcal{D}$ is required in order to conduct the approximation in Algorithm 4. Given an error bound ϵ, the bound condition is then fixed. In order to minimize k, we expect that the P values of the tuples from $k + 1$ to n in $\overline{B}(k) = \sum_{j=k+1}^{n} s_j[P]$ are small. In other words, an instance of $\mathcal{D}$ with higher P in the tuples from 1 to k is preferred. Therefore, we can heuristically reorganize the tuples in $\mathcal{D}$ in the decreasing order of P as the input of Algorithm 4. The ordering of statistical tuples in $\mathcal{D}$ by the P values can be done in linear time by amortizing the P values into a constant domain.

5.3. Approximation Individual

Next, we study the approximation in each individual candidate λ_j with a more efficient bound condition respectively. According to formula (3) in the proof of error bound, we find that for each specific candidate λ_j if

Algorithm 4. Approximation Algorithm

AP $(\mathcal{D}, \mathcal{C}_t, \lambda_Y)$

Algorithm 4 Approximation Algorithm
AP$(\mathcal{D}, C_t, \lambda_Y)$

Input: Statistical distribution $\mathcal{D}$, Candidate set C_t, and λ_Y
Output: All the valid λ_j

1: **for** each tuple $s_k \in \mathcal{D}, k : n \to 1$ **do**
2: $\overline{B}$ += $s_k[P]$
3: **if** $\overline{B} > \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$ **then**
4: k++; **break** {Compute k}
5: **end if**
6: **end for**
7: **for** each candidate $\lambda_j \in C_t, j : 1 \to c$ **do**
8: $P_0^j(X, Y) = P_0^j(X) = 0$
9: **for** each tuple $s_i \in \mathcal{D}, i : 1 \to k$ **do**
10: compute $P_i^j(X)$ and $P_i^j(X, Y)$ for λ_j, λ_Y
11: **end for**
12: **end for**
13: **return** all the λ_j with confidence and support satisfying η_c, η_s

$$\overline{\beta} \le \min\left(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}\right), \tag{5}$$

then the error bound is already satisfied and the processing can be terminated for this λ_j. Therefore, rather than one fixed bound condition for all the candidates, the bound of $\overline{\beta}$ can be determined dynamically for each candidate λ_j respectively during the processing.

Algorithm 5 shows the implementation of approximation with dynamic bound condition on each candidate λ_j individually. In particular, Lines 14–16 accomplish the possible early termination for each individual λ_j according to the condition discussed in formula (5).

Corollary 1. *The worst case complexity of the Approximation Individual is $\mathcal{O}(kc)$.*

Proof. Note that with the increase of i from 1 to k, for a specific λ_j, the value β increases and $\overline{B}_j$ decreases. For any $i < k$, if $\beta < \eta_s$, i.e., λ_j is invalid currently, the bound condition cannot be satisfied having

$$\min\left(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}\right) < \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right) < \overline{B}_j.$$

When λ_j has $\beta \geq \eta_s$ as a valid threshold, the bound condition is relaxed from $min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right)$ to $min\left(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}\right)$. Thereby, the bound condition may be satisfied by a smaller i than k, i.e.,

$$\min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right) < \overline{B}_j \leq \min\left(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}\right).$$

The worst case is that all candidates do not achieve their bounds until processing the tuple s_k, where

$$\overline{B}_j = \overline{B}(k) \leq \min\left(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}\right) \leq \min\left(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}\right)$$

must be satisfied. This is exactly the Algorithm 4 without individual approximation. □

Finally, we cooperate the pruning of candidates by support together with the approximation (namely APS) and the approximation individual (namely APSI) respectively. As we presented in the experimental evaluation, the approximation techniques can further improve the discovering efficiency with an approximate solution very close to the exact one (bounded by ϵ).

Algorithm 5. Approximation Individual

Algorithm 5 Approximation Individual
API$(\mathcal{D}, C_t, \lambda_Y)$

Input: Statistical distribution $\mathcal{D}$, Candidate set C_t, and λ_Y
Output: All the valid λ_j

1: **for** each tuple $s_i \in \mathcal{D}, i : n \to 1$ **do**
2: $\bar{B}$ += $s_i[P]$
3: **if** $\bar{B} \leq \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$ **then**
4: $k = i$ {Compute k}
5: **end if**
6: **end for**
7: **for** each candidate $\lambda_j \in C_t, j : 1 \to c$ **do**
8: $P_0^j(X, Y) = P_0^j(X) = 0$
9: $\bar{B}_j = \bar{B}$
10: **for** each tuple $s_i \in \mathcal{D}, i : 1 \to k$ **do**
11: compute $P_i^j(X)$ and $P_i^j(X, Y)$ for λ_j, λ_Y
12: $\beta = P_i^j(X)$
13: $\bar{B}_j$ -= $s_i[P]$
14: **if** $\bar{B}_j \leq \min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c})$ **then**
15: **break**
16: **end if**
17: **end for**
18: **end for**
19: **return** all the λ_j with confidence and support satisfying η_c, η_s

6. Experimental evaluation

Now, we report the experiment evaluation on the proposed methods. All the algorithms are implemented using Java. The experiments run on a machine with Intel Core 2 CPU (2.13 GHz) and 2 GB of memory.

6.1. Experiment setting

We use three real data sets in the experimental evaluation. The *Cora*[2] data set, prepared by McCallum et al. [38], consists of 1296 records of scientific papers with attributes author, volume, title, institution, venue, etc. The *Restaurant*[3] data set consists of 865 restaurant records including attributes name, address, city and type. The *CiteSeer*[4] data set selects 10,000 tuples with attributes title, author, address, affiliation, subject, description, etc. In the off-line pre-processing, we use the *cosine* similarity to evaluate the matching quality of the tuples in the original data. By applying the sim(A) mapping in Section 3, we can obtain statistical distributions with at most 186,031 statistical tuples in *Cora*, 140,781 statistical tuples in *Restaurant* and 314,382 statistical tuples in *CiteSeer*. Consequently, in experimental evaluation, the inputs of algorithms are

[2] http://www.cs.umass.edu/mccallum/code-data.html.
[3] http://www.cs.utexas.edu/users/ml/riddle/data.html.
[4] http://citeseer.ist.psu.edu/.

the off-line pre-processed statistical distributions with various data sizes, i.e., having statistical tuples n from 10,000 to 150,000 respectively.

We mainly study the efficiency of the proposed algorithms. Since our main task is to discover MDs under the required η_s and η_c, we study the runtime performance in various distributions with different η_s and η_c settings. The discovery algorithms determine the similarity threshold settings of attributes for MDs. Suppose that users want to discover MDs on the following $X \rightarrow Y$ of three data sets respectively: i) the dependencies on

$$Cora : \text{author}, \text{volume}, \text{title} \rightarrow \text{venue}$$

with the preliminary requirement of minimum similarity 0.6 on venue; ii) the dependencies on

$$Restaurant : \text{name}, \text{address}, \text{type} \rightarrow \text{city}$$

with the preliminary requirement of minimum similarity 0.5 on city; and iii) the dependencies on

$$CiteSeer : \text{address}, \text{affiliation}, \text{description} \rightarrow \text{subject}$$

with preliminary minimum similarity 0.1 on subject, respectively.

A returned result is either infeasible, or an MD with similarity threshold pattern on the given $X \rightarrow Y$. For example, one of the results returned by the experiments on *Cora* is:

$$\varphi(\text{author}, \text{volume}, \text{title} \rightarrow \text{venue}, <0.6, 0.0, 0.8, 0.6>)$$

with support(φ) = 0.020 and confidence(φ) = 0.562 both greater than the specified requirements of η_s and η_c respectively. Remarkably, the similarity threshold of attribute volume is 0.0 in the result, which is exactly the case discussed in Section 4.1. It states that the similarity of volume has no effect on venue and can be removed from the dependency. This result verifies that our discovery process can automatically identify those attributes that are not required in X for an MD φ.

6.2. Exact approach evaluation

In the first experiment, we evaluate the performance of pruning by support (EPS) compared with the original exact algorithm (EA). Figs. 1, 2 and 3 report the time cost over three data sets respectively, given different η_s and η_c requirements. As shown in the

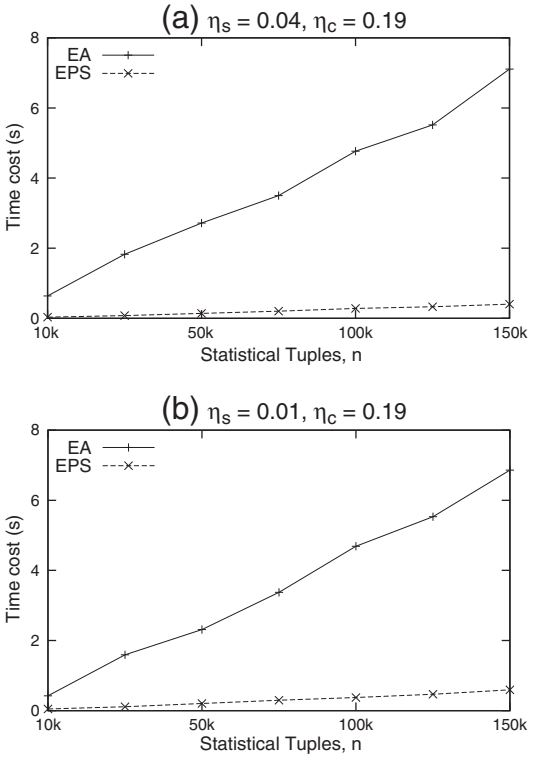


Fig. 1. A pruning on *CiteSeer*.

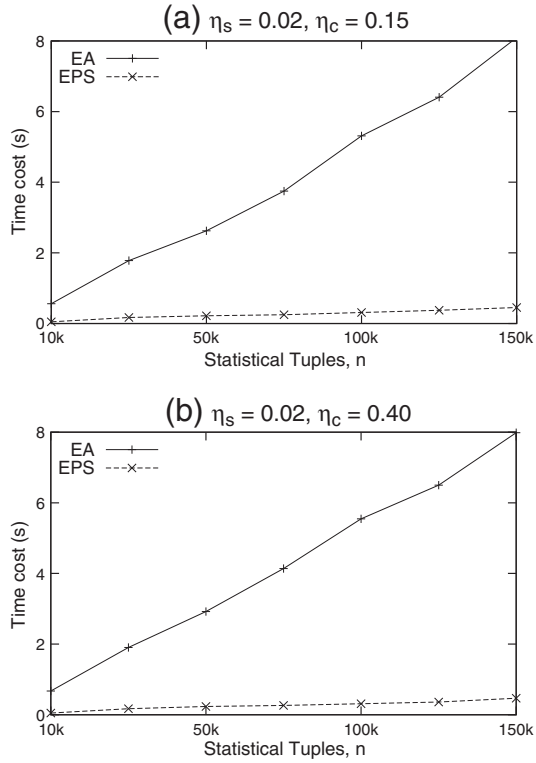


Fig. 2. A pruning on *Cora*.

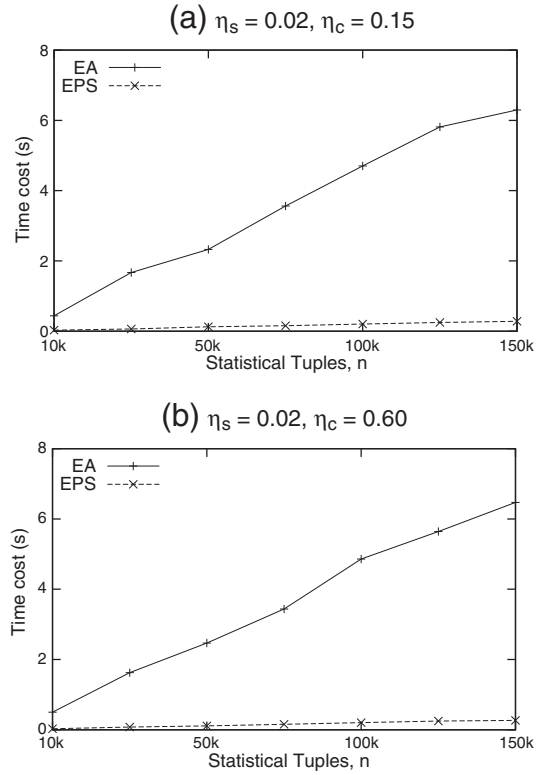


Fig. 3. A pruning on *Restaurant*.

sub-figures (a) and (b), the EA method, which has to evaluate all the possible candidates, should have the same cost no matter how η_s and η_c are set. Therefore, the time cost of EA in (a) is exactly the same as that in (b) in all three data sets.

Moreover, the EPS achieves significantly lower time cost in all the statistical distributions, which is only about 1/10 of that of the EA. These results demonstrate that our EPS approach can prune most of the candidates without costly computation. Note that the time costs of approaches increase linearly with the sizes of statistical distributions, which shows the scalability of discovering MDS on large data.

To observe more accurately, we also plot the EPS time cost in Figs. 4, 5 and 6 with the same settings respectively. According to the pruning strategy, the EPS performance is only affected by support requirement η_s. In other words, different η_c settings take no effect on EPS. Thus, EPS has similar time costs in Fig. 5(a) and (b) with the same η_s but different η_c. Similar results can be observed in Fig. 6 as well.

On the other hand, the EPS approach conducts the pruning based on the given requirement of support η_s. It is natural that a higher η_s turns to the better pruning performance. Therefore, EPS with $\eta_s = 0.04$ in Fig. 4(a) shows lower time cost, e.g., about 0.4 s for 150 k, than that of $\eta_s = 0.01$ in (b), e.g., 0.6 s for the same 150 k. Similar results with different η_s are also observed and omitted on *Cora* and *Restaurant*.

6.3. Advanced approach evaluation

Now, we report the performance of advanced pruning and approximation techniques in Figs. 4, 5 and 6, including the pruning by both support and confidence (EPSC), the approximation together with pruning by support (APS), and the approximation individual together with pruning by support (APSI).

First, we study the influence of η_c in different approaches. When the confidence requirement η_c is high, e.g., in Figs. 5(b) and 6(b), the EPSC can remove those low confidence candidates and show better time performance than other approaches. On the other hand, when η_c is small, e.g., $\eta_c = 0.15$, we can have larger choices of $\epsilon \in (0, 1 - \eta_c)$ such as $\epsilon = 0.8 \in (0, 1 - 0.15)$ in Figs. 5(a) and 6(a). Thus, the approximation approaches have lower time cost, especially the APSI. According to this analysis, we can choose EPSC in practical cases if the requirement η_c is high; otherwise, the APSI is preferred in order to achieve lower time costs.

According to the bound condition of approximation approaches in Theorem 1, not only ϵ, but also the η_s affects the performance. As presented in Fig. 4(a), a higher η_s contributes a larger bound condition, which means the early termination of

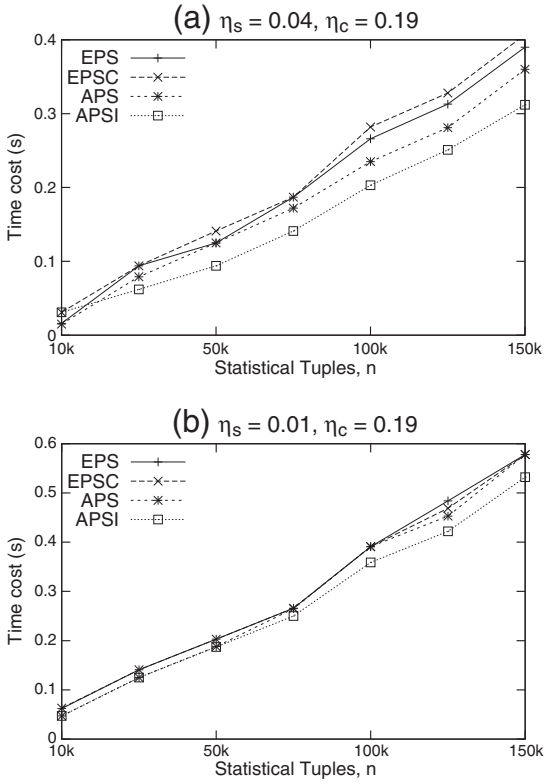


Fig. 4. Advanced approaches on *CiteSeer*.

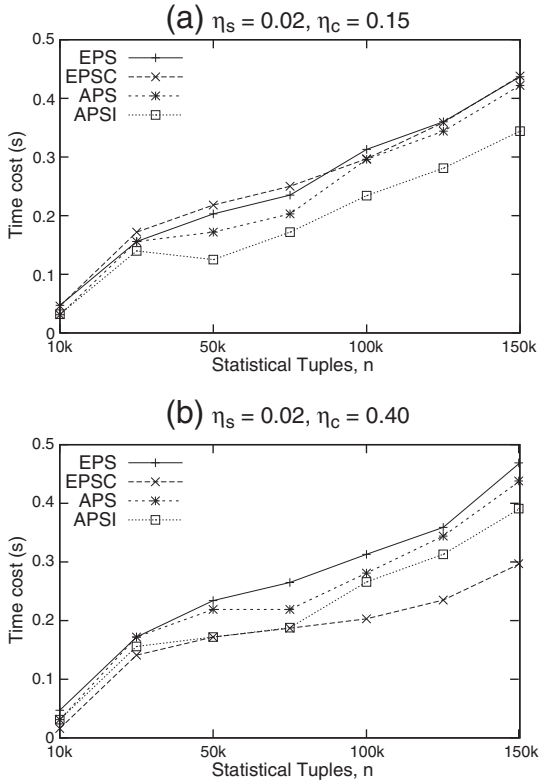


Fig. 5. Advanced approaches on *Cora*.

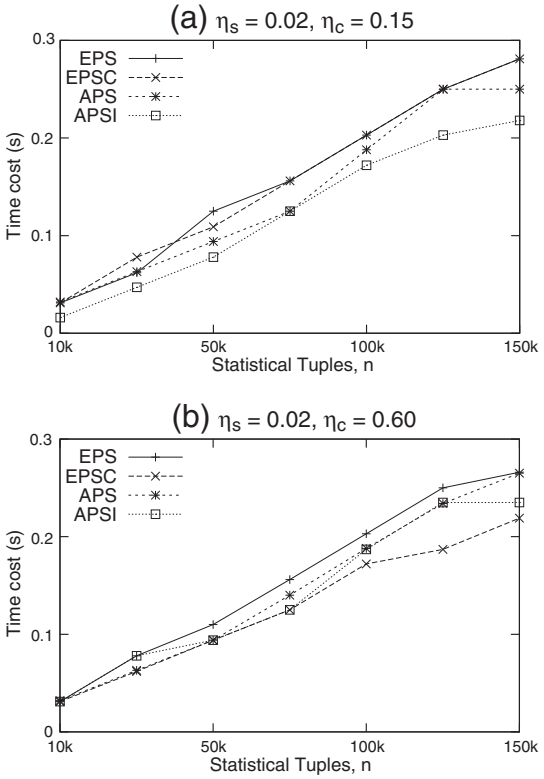


Fig. 6. Advanced approaches on *Restaurant*.

the program. Thus, approximation approaches show better performance in Fig. 4(a), having $\eta_s = 0.04$, compared with Fig. 4(b), whose $\eta_s = 0.01$.

Note that the bound condition also depends on the distribution features. A preferred distribution with more tuples in $\overline{\beta}$ can achieve the bound condition and terminate early, such as 50 k in Fig. 5(a) with low time cost.

Finally, we evaluate the approximate confidence and support of the returned MDs with $\epsilon = 0.8$ on all the data sets in Figs. 7, 8 and 9. First, we can observe that the exact approaches EA, EPS and EPSC return the same measures. These results verify the correctness of our exact computation methods. Moreover, according to Lemma 3, the error introduced in approximation approaches is bounded by ϵ on both confidence and support. Therefore, as illustrated in Figs. 7, 8 and 9, the approximate confidence and support of APS and APSI are very close to those of exact algorithms. In particular, the approximation methods show good approximation performance as well in larger data sizes, in all the three data sets.

Consequently, the approximation algorithm can achieve low time cost (e.g., in Figs. 4(a), 5(a) and 6(a) with the same setting of ϵ) without introducing a large variation in the confidence and support measures compared with the exact ones.

6.4. Summary

The experiment results demonstrate that our pruning and approximation techniques can significantly improve the efficiency of discovering MDs.

- The time costs of approaches increase linearly with the sizes of statistical distributions, which shows the scalability of discovering MDs on large data.
- The EPS approach can significantly reduce the time costs by pruning candidates, compared with the EA.
- If the minimum confidence requirement η_c is high, the pruning by confidence works well.
- Otherwise, we can employ the approximation approaches to achieve low time cost.

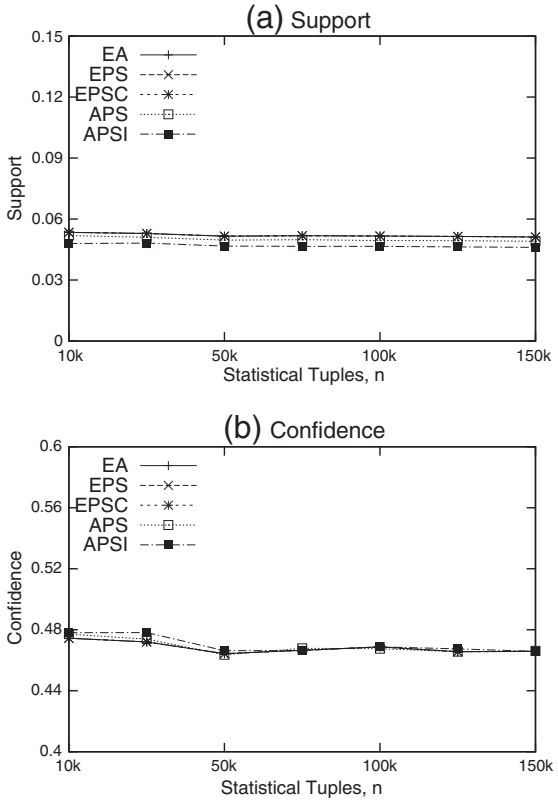


Fig. 7. Relative error of approximate support and confidence on *CiteSeer*.

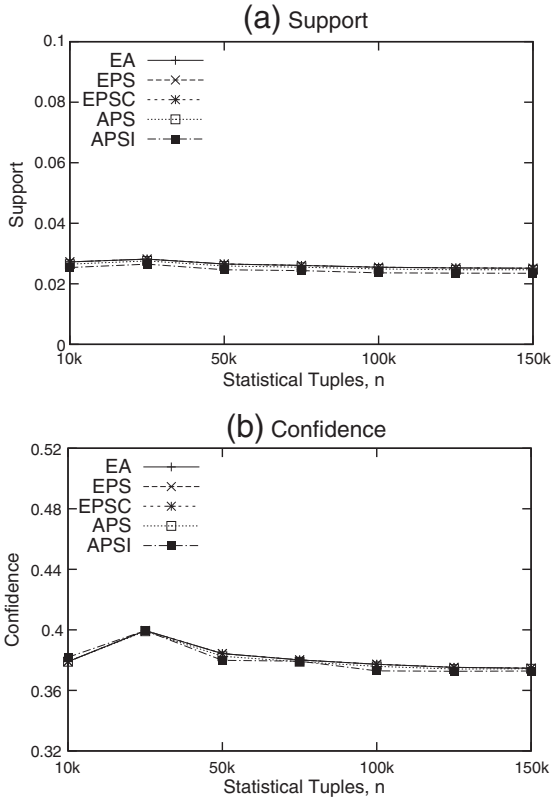


Fig. 8. Relative error of approximate support and confidence on *Cora*.

7. Conclusions

In this paper, we study the efficient discovery of matching dependencies. First, we introduce the utility evaluation of matching dependencies by using support and confidence. Then, we formalize the problem of discovering MDs with the minimum confidence and support requirements. Both pruning strategies and approximation of the exact algorithms are studied. The pruning by support can filter out the candidate patterns with low supports. In addition, if the minimum confidence requirement is high, the pruning by confidence works well; otherwise, we can employ the approximation approaches to achieve low time cost. The experimental evaluation demonstrates the performance of the proposed methods.

There are many aspects of work to develop in the future, for discovering matching dependencies. For example, although the current approach can exclude the attributes that are not necessary to an MD, another issue is to minimize the number of attributes in the MD. However, the problem of determining attributes for FDs is already hard [29], where the matching similarity thresholds are not necessary to be considered. Moreover, two different MDs may cover different dependency semantics, which leads us to the problem of generating MDs set. Rather than a single MD, the utility evaluation of an MDs set is also interesting. Most importantly, more exciting applications of MDs are expected to be explored in the future work. Finally, it is also interesting to introduce similarity constraints to Inclusion and Multivalued Dependencies. For instance, a novel type of dependencies, say Metric Inclusion Dependencies, may specify the constraint that one relation must contain tuples similar to the tuples in another relation. The discovery of similarity constraints for such novel dependencies is also promising.

Acknowledgments

The work described in this paper was partially supported by China NSFC Project No. 61202008; Hong Kong RGC GRF Project No. 611411; National Grand Fundamental Research 973 Program of China under Grant 2012CB316200; Microsoft Research Asia Grant and Huawei Noahs ark lab grant HWLB06-15C03212/13PN.

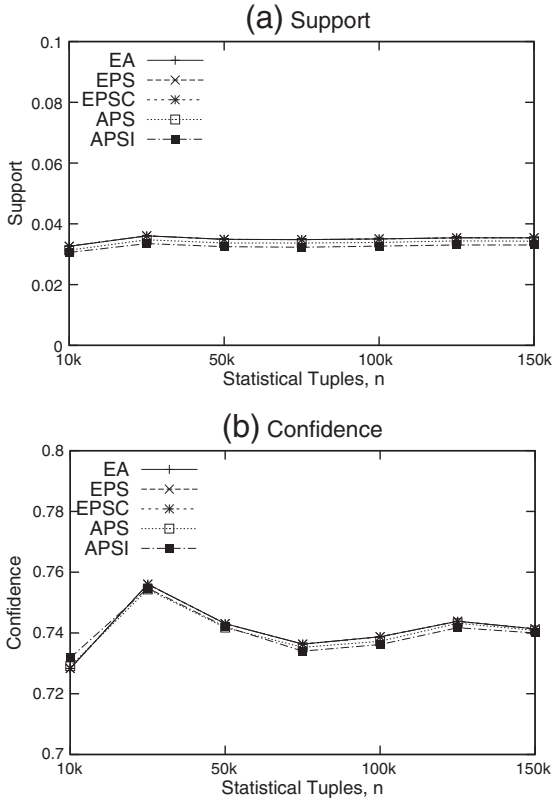


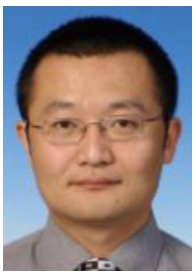
Fig. 9. Relative error of approximate support and confidence on *Restaurant*.

References

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.
[2] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, SIGMOD Conference, 1993, pp. 207–216.
[3] R. Bassée, J. Wijsen, Neighborhood dependencies for prediction, PAKDD, 2001, pp. 562–567.
[4] C. Batini, M. Scannapieco, Data quality: concepts, methodologies and techniques, Data-Centric Systems and Applications, Springer, 2006.
[5] L.E. Bertossi, S. Kolahi, L.V.S. Lakshmanan, Data cleaning and query answering with matching dependencies and matching functions, ICDT, 2011, pp. 268–279.
[6] M. Bilenko, R.J. Mooney, W.W. Cohen, P. Ravikumar, S.E. Fienberg, Adaptive name matching in information integration, IEEE Intelligent Systems 18 (5) (2003) 16–23.
[7] D. Bitton, J. Millman, S. Torgersen, A feasibility and performance study of dependency inference, ICDE, 1989, pp. 635–641.
[8] P. Bohannon, W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for data cleaning, ICDE, 2007, pp. 746–755.
[9] L. Bravo, W. Fan, F. Geerts, S. Ma, Increasing the expressivity of conditional functional dependencies without extra complexity, ICDE, 2008, pp. 516–525.
[10] L. Bravo, W. Fan, S. Ma, Extending dependencies with conditions, VLDB, 2007, pp. 243–254.
[11] T. Calders, R.T. Ng, J. Wijsen, Searching for dependencies at multiple abstraction levels, ACM Transactions on Database Systems 27 (3) (2002) 229–260.
[12] F. Chiang, R.J. Miller, Discovering data quality rules, PVLDB 1 (1) (2008) 1166–1177.
[13] W.W. Cohen, Integration of heterogeneous databases without common domains using queries based on textual similarity, SIGMOD Conference, 1998, pp. 201–212.
[14] G. Cong, W. Fan, F. Geerts, X. Jia, S. Ma, Improving data quality: consistency and accuracy, VLDB, 2007, pp. 315–326.
[15] A.K. Elmagarmid, P.G. Ipeirotis, V.S. Verykios, Duplicate record detection: a survey, IEEE Transactions on Knowledge and Data Engineering 19 (1) (2007) 1–16.
[16] W. Fan, Dependencies revisited for improving data quality, PODS, 2008, pp. 159–170.
[17] W. Fan, H. Gao, X. Jia, J. Li, S. Ma, Dynamic constraints for record matching, The VLDB Journal (2010) 1–26.
[18] W. Fan, F. Geerts, L.V.S. Lakshmanan, M. Xiong, Discovering conditional functional dependencies, ICDE, 2009, pp. 1231–1234.
[19] W. Fan, J. Li, X. Jia, S. Ma, Reasoning about record matching rules, PVLDB, 2009.
[20] W. Fan, J. Li, S. Ma, N. Tang, W. Yu, Interaction between record matching and data repairing, SIGMOD Conference, 2011, pp. 469–480.
[21] W. Fan, S. Ma, Y. Hu, J. Liu, Y. Wu, Propagating functional dependencies with conditions, PVLDB 1 (1) (2008) 391–407.
[22] P.A. Flach, I. Savnik, Database dependency discovery: a machine learning approach, AI Communications 12 (3) (1999) 139–160.
[23] J. Gardezi, L.E. Bertossi, I. Kiringa, Matching dependencies with arbitrary attribute values: semantics, query answering and integrity constraints, LID, 2011, pp. 23–30.
[24] C. Giannella, E.L. Robertson, On approximation measures for functional dependencies, Information Systems 29 (6) (2004) 483–507.
[25] L. Golab, H.J. Karloff, F. Korn, A. Saha, D. Srivastava, Sequential dependencies, PVLDB 2 (1) (2009) 574–585.
[26] L. Golab, H.J. Karloff, F. Korn, D. Srivastava, B. Yu, On generating near-optimal tableaux for conditional functional dependencies, PVLDB 1 (1) (2008) 376–390.
[27] L. Gravano, P.G. Ipeirotis, N. Koudas, D. Srivastava, Text joins in an rdbms for web data integration, WWW, 2003, pp. 90–101.
[28] Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen, Efficient discovery of functional and approximate dependencies using partitions, ICDE, 1998, pp. 392–401.
[29] Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen, Tane: an efficient algorithm for discovering functional and approximate dependencies, The Computer Journal 42 (2) (1999) 100–111.

[30] I.F. Ilyas, V. Markl, P.J. Haas, P. Brown, A. Aboulnaga, Cords: automatic discovery of correlations and soft functional dependencies, SIGMOD Conference, 2004, pp. 647–658.
[31] R.S. King, J.J. Legendre, Discovery of functional and approximate functional dependencies in relational databases, JAMDS 7 (1) (2003) 49–59.
[32] J. Kivinen, H. Mannila, Approximate inference of functional dependencies from relations, Theoretical Computer Science 149 (1) (1995) 129–149.
[33] N. Koudas, A. Saha, D. Srivastava, S. Venkatasubramanian, Metric functional dependencies, ICDE, 2009, pp. 1275–1278.
[34] S. Kramer, B. Pfahringer, Efficient search for strong partial determinations, KDD, 1996, pp. 371–374.
[35] S.E. Madnick, H. Zhu, Improving data quality through effective use of data semantics, Data & Knowledge Engineering 59 (2) (2006) 460–475.
[36] H. Mannila, K.-J. Räihä, Design of Relational Databases, Addison-Wesley, 1992.
[37] H. Mannila, K.-J. Räihä, Algorithms for inferring functional dependencies from relations, Data & Knowledge Engineering 12 (1) (1994) 83–99.
[38] A. McCallum, K. Nigam, L.H. Ungar, Efficient clustering of high-dimensional data sets with application to reference matching, KDD, 2000, pp. 169–178.
[39] G. Navarro, A guided tour to approximate string matching, ACM Computing Surveys 33 (1) (2001) 31–88.
[40] B. Pfahringer, S. Kramer, Compression-based evaluation of partial determinations, KDD, 1995, pp. 234–239.
[41] T. Scheffer, Finding association rules that trade support optimally against confidence, Intelligent Data Analysis 9 (4) (2005) 381–395.
[42] J.C. Schlimmer, Efficiently inducing determinations: a complete and systematic search algorithm that uses optimal pruning, ICML, 1993, pp. 284–290.
[43] S. Song, L. Chen, Discovering matching dependencies, CIKM, 2009, pp. 1421–1424.
[44] S. Song, L. Chen, Differential dependencies: reasoning and discovery, ACM Transactions on Database Systems 36 (4) (2011).
[45] S. Song, L. Chen, P.S. Yu, On data dependencies in dataspaces, ICDE, 2011, pp. 470–481.
[46] U. ul Hassan, S. O'Riain, E. Curry, Leveraging matching dependencies for guided user feedback in linked data applications, Proceedings of the Ninth International Workshop on Information Integration on the Web, IIWeb '12, ACM, New York, NY, USA, 2012, pp. 5:1–5:6.
[47] H. Wang, R. Liu, Privacy-preserving publishing microdata with full functional dependencies, Data & Knowledge Engineering 70 (3) (2011) 249–268.
[48] C.M. Wyss, C. Giannella, E.L. Robertson, Fastfds: a heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances — extended abstract, DaWaK, 2001, pp. 101–110.

Shaoxu Song is currently an assistant professor in the Tsinghua National Laboratory for Information Science and Technology; School of Software, Tsinghua University, China. He received his PhD degree in computer science from the Hong Kong University of Science and Technology. His research interests include data quality and data dependency.



Lei Chen is currently an associate professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. He received his BS degree in computer science and engineering from Tianjin University, China, in 1994, the MA degree from Asian Institute of Technology, Thailand, in 1997, and the PhD degree in computer science from University of Waterloo, Canada, in 2005. He served as an associate editor of the IEEE Transactions on Knowledge and Data Engineering. His research interests include uncertain databases, graph databases, multimedia and time series databases, and sensor and peer-to-peer databases.