Discovering Matching Dependencies

Shaoxu Song
Hong Kong University of Science and Technology, Hong Kong
sshaoxu@cs.ust.hk

Lei Chen
Hong Kong University of Science and Technology, Hong Kong
leichen@cs.ust.hk

ABSTRACT

The concept of *matching dependencies* (MDs) is recently 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 violation detection. In this paper, we first formally define the statistical measures to evaluate MDs in a given database instance. Then, we study the problem of discovering MDs with certain *similarity threshold* settings on attributes. Moreover, since MDs might not be able to express many matching rules, we propose the *extended matching dependencies* (eMDs) to capture the dependencies through a set of patterns with matching *similarity intervals* in a pattern tableau. During the discovery of eMDs, it is naturally desirable to find the most concise pattern tableau that can still satisfy the user requirements. Unfortunately, as we proved, the minimal pattern tableau problem is NP-complete. Therefore, we study the greedy algorithm to discover near optimal eMDs and propose pruning techniques to further improve the discovery performance. Finally, our experimental evaluation demonstrates the efficiency of the proposed methods.

1. INTRODUCTION

Recently, data quality has become a hot topic in database community due to huge amount of "dirty" data originated from different resources (see [2] for a survey). These data are often "dirty", including inconsistencies, conflicts, and errors, due to various erroneous introduced by human and machines. In addition to cost of dealing 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. For example, consider a Contacts relation with the schema:

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

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) [4], i.e., FDs with conditions, which have more expressive power. The basic idea of these extensions is making the FDs, originally hold for the whole table, valid only for a set of tuples. For example, the following cfd specifies that only in the condition of country code $\mathsf{CC} = 44$, if two tuples have the same ZIP, then they must have same Street as well.

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

These dependency constraints can be used to detect data violations [10]. For instance, we can use the above fd to detect violations in an instance of Contacts in Table 1. For the tuples t_5 and t_6 with the same values of $\mathsf{ZIP} = 021$, they have different values of City, which are then 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 [10], they check the specified attribute value agreement based on *exact match*. For example, with the above cfd, tuples that have $\mathsf{CC} = 44$ and the same value on ZIP attribute will be checked to see whether they have exactly matched values on Street. Obviously, this strict exact match constraint limits usage of FDs and CFDs, since real-world information often have various representation formats. For example, the tuples t_2 and t_3 in Contacts table will be detected as "violations" of the cfd, since they have "different" Street values but agree on ZIP and $\mathsf{CC} = 44$. However, "No.2, Central Rd." and "#2, Central Rd." are exactly the "same" street in the real-world with different representation formats.

To make dependencies adapt to this real-world scenario, i.e., same information have different representation formats, Fan [12] proposed a new concept of data dependencies, called *matching dependencies* (MDs). Informally, a matching dependency targets on the fuzzy values like text attributes and defines the dependency between two set of attributes according to their matching quality measured by some matching operators (see [3] for a survey), such as *Euclidean distance* and *cosine similarity*. Again, in Contacts example, we may have a MD as

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

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).

Similar to the FDs related techniques, MDs can be applied in many tasks as well. For example, in **data cleaning**, we can also use MDs to detect the inconsistent data, that is, data do not follow the constraint (rule) specified by MDs. For example, base on the

VLDB '09, August 24-28, 2009, Lyon, France

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

above md_1 example, we search the pairs of tuples whose similarities on Street are greater than 0.8 while the City similarities are lower than 0.7. In addition to locating the inconsistent data, **object identification**, another important work for data cleaning, can also employ MDs as matching rules. For instance, according to

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

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.

Though the concept of matching dependencies is given in [12], the authors did not discuss how to discover 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, 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 a MD $([\mathsf{ZIP}] \rightarrow [\mathsf{City}], < 1.0, 1.0 >)$. Clearly, not all the settings of thresholds for MDs are useful.

Therefore, the first question that we have to address is how to measure the "usefulness" of a detected MD. In this work, we adopt the widely used measures *confidence* and *support*. Specifically, we consider a 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 $\mathsf{support}(\varphi)$ measures the *probability* that the matching similarities of any pair of tuples t_1 and t_2 on attributes X and Y satisfy the corresponding thresholds in pattern λ. The $\mathsf{confidence}(\varphi)$ is the *conditional probability* of t_1 and t_2 with matching similarity on Y satisfying λ_Y given the condition that t_1 and t_2 are similar on attributes X (satisfying λ_X). Clearly, in real applications, such as object identification, users often seek useful matching rules with high support and confidence to identify the duplicate objects. Therefore, in this work, we would like to discover proper settings of matching similarity thresholds for MDs, which can satisfy users' requirements (i.e. users specified support and confidence).

During the discovery of MDs, we also find that the expressive power of MDs by setting up single similarity thresholds is not enough to capture many useful matching rules. For example, we consider the above md_2 again, for the object identification problem according to Name and Street attributes. The matching similarity of Jason Smith and J.Smith is 0.6, which is lower than that of J.Smith and W.J.Smith (i.e. 0.85). In fact, however, Jason Smith and J.Smith (low similarity) denote the same person, while J.Smith and W.J.Smith (high similarity) are not (i.e., having different SIN). Therefore, we cannot address all the mentioned matching rules by setting a single matching similarity threshold, such as $\lambda[\mathsf{Name}] = 0.6$ or 0.85.

To address this limited expressive issue, in this work, we introduce *extended matching dependencies* (eMDs), which specifies the dependencies by matching similarity intervals, rather than single similarity thresholds in MDs. For example, the interval $[0.6, 0.8)$ describes the constraint of matching similarity between 0.6 and 0.8. Consequently, the following emd_1 specifies the dependency that for any two tuples, if their matching similarities are between $[0.6, 0.8)$ on attribute Name, and between $[0.9, 1.0]$ on attribute Street, then they must have the same SIN.

$$\mathsf{emd}_1 : ([\mathsf{Name}, \mathsf{Street}] \rightarrow [\mathsf{SIN}], < [0.6, 0.8), [0.9, 1.0], 1.0 >)$$
$$\mathsf{emd}_2 : ([\mathsf{Name}, \mathsf{Street}] \rightarrow [\mathsf{SIN}], < [0.9, 1.0], [0.9, 1.0], 1.0 >)$$

More than one matching similarity intervals may be valid for an attribute. For instance, emd_2 specifies another interval $[0.9, 1.0]$ on Name for the dependencies on the same attributes. We use a tableau to represent the patterns of intervals in eMDs, instead of a single threshold pattern in MDs. An eMD is denoted as $\psi(X \rightarrow Y, \mathcal{T})$, where $\mathcal{T}$ is a pattern tabular which specifies a set of interval patterns on X and Y. For the above Name, Street, and SIN dependency, we can have an eMD: $([\mathsf{Name}, \mathsf{Street}] \rightarrow [\mathsf{SIN}], \mathcal{T})$, where $\mathcal{T}$ is:

Name	Street	SIN
$[0.6, 0.8)$	$[0.9, 1.0]$	1.0
$[0.9, 1.0]$	$[0.9, 1.0]$	1.0

The same as MDs, eMDs can also be applied in violation detection. Those pairs of tuples are detected as violations of an eMD where some of the interval patterns of the eMD are violated. Thus, the patterns of intervals in the tableau are frequently evaluated during the violation detection. However, two patterns in an arbitrary pattern tableau may cover the same semantics of intervals, such as $[0.6, 0.7)$ and $[0.6, 0.8)$, i.e., redundancy. Thus, the more concise the pattern tableau is, the better the detection efficiency is. This motivation introduces an interesting optimization problem, to discover concise eMDs with the minimum number of interval patterns.

Contributions. In this paper, given a relation instance, we study the issues of discovering the matching dependencies. Our main contributions are summarized as follows:

First, we propose the formal definition of *extended matching dependencies* (in Section 3). The corresponding confidence and support evaluations of eMDs are developed as well. To the best of our knowledge, this is the first paper to propose and evaluate eMDs.

Second, we study the discovery of MDs (in Section 4). The MDs discovery problem is to find settings of matching similarity thresholds on the attributes for MDs that can satisfy the required confidence and support. We first present an exact algorithm and study pruning strategies by the minimum requirements of support and confidence. In addition, to avoid the traversal of all the data, we propose an approximate solution with bounded error.

Third, we study the discovery of eMDs (in Section 5). As we proved, the problem of discovering optimal eMDs with the minimum number of interval patterns is NP-complete. Therefore, we introduce greedy algorithm to find approximate optimal solutions with an error bound on tableau size. Moreover, to improve the efficiency, pruning strategies are proposed to filter out unqualified candidate patterns, with a proved bound of the pruning rate.

Finally, we report an extensive experimental evaluation (in Section 6). The proposed algorithms on discovering MDs and eMDs are studied. Our pruning strategies can significantly improve the efficiency in discovering MDs and eMDs.

All the proofs can be found in the full version [15] of this paper. Table 2 lists the frequently used notations in this paper.

2. RELATED WORK

Recently, traditional 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. The conditional functional dependencies (CFDs) are first

Table 2: Notations

Symbol	Description
φ	Matching dependency, MD
ψ	Extended matching dependency, eMD
λ	Threshold pattern, of matching similarity thresholds
δ	Interval pattern, of matching similarity intervals
$\mathcal{T}$	Tableau, of interval patterns
$\mathcal{C}_t$	Candidate set, of threshold patterns
$\mathcal{C}_e$	Candidate set, of interval patterns
η_s	Minimum requirement, of support
η_c	Minimum requirement, of confidence
$\mathcal{R}$	Original relation, of N data tuples t
$\mathcal{D}$	Statistical distribution, of n statistical tuples s

proposed in [4] for data cleaning. Cong et al. [10] study the detecting and repairing methods of violation by CFDs. Fan et al. [14] investigate the propagation of CFDs for data integration. Bravo et al. [5] propose an extension of CFDs by employing disjunction and negation. Golab et al. [17] define a range tableau for CFDs, where each value is a range similar to the concept of matching similarity intervals in our study. In addition, Bravo et al. [6] propose conditional inclusion dependency (CINDs), which are useful not only in data cleaning, but are also in contextual schema matching.

Inspired by the above interesting applications, the discovery of CFDs is studied as well. The confidence and support measures are widely used in evaluating CFDs [17, 7, 13]. In addition, Chiang and Miller [7] also study some other measures such as conviction and χ^2-test. When a candidate $X \rightarrow Y$ is suggested together with minimum support and confidence, Golab et al. [17] study the discovery of optimal CFDs with the minimum pattern tableau size. A concise set of patterns are naturally desirable which may have lower cost during the applications such as violation detection by CFDs. Our optimal eMD problem is based on the same principle. On the other hand, Chiang and Miller [7] explore CFDs by considering all the possible dependency candidates when $X \rightarrow Y$ is not specified. In [13], Fan et al. also study the case when the embedded FDs are not given, and propose three algorithms for different scenarios.

The concept of matching dependencies (MDs) is first proposed in [12] for specifying matching rules for the object identification (see [11] for a survey). The MDs can be regarded as a generalization of FDs, which are based on identical values having matching similarity equal to 1.0 exactly. Thus, FDs can be represented by the syntax of MDs as well. For any two tuples, if their X values are identical (with similarity ≥ 1.0), then a FD ($X \rightarrow Y$) requires that their Y values are identical too, i.e., a MD ($X \rightarrow Y, < 1.0, 1.0 >$). Fan [12] gives the concept of matching dependencies without introducing how to evaluate and discover MDs.

3. STATISTICAL EVALUATION

In this section, we formally introduce the definitions of MDs and eMDs, respectively. Then, we develop statistical analysis for evaluating MDs and eMDs over a given database instance.

3.1 Matching Dependencies

Traditional functional dependencies FDs and their extensions rely on the exact matching operator $=$ to identify dependency relationships. However, in the real world application, it is not possible to use exact matching operator $=$ to identify matching over fuzzy data values such as text values. For instance, Jason Smith and J.Smith of attribute Name may refer to the same real world entity. Therefore, instead of FDs on identical values, the *matching dependencies* MDs [12] are proposed based on the matching quality. For text values, we can adopt the similarity matching operators, denoted by $\approx$, such as *edit distance* [21], *cosine similarity* with word tokens [9] or *q-grams* [18].

Consider a relation $\mathcal{R}(A_1, \ldots, A_M)$ with M attributes. Following similar syntax of FDs, we define MDs as following: [1]

DEFINITION 1. *A* matching dependency (MD) φ *is a pair* $(X \rightarrow Y, \lambda)$*, where* $X \subseteq \mathcal{R}, Y \subseteq \mathcal{R}$*, and* λ *is a* threshold pattern *of matching similarity thresholds on attributes in* $X \cup Y$*, e.g.,* $\lambda[A]$ *denotes the matching similarity threshold on attribute* A*.*

A MD φ specifies a constraint on the set of attributes X to Y. Specifically, the constraint states that, for any two tuples t_1 and t_2 in a relation instance r of $\mathcal{R}$, if $\bigwedge_{A_i \in X} t_1[A_i] \approx_{\lambda[A_i]} t_2[A_i]$, then $\bigwedge_{A_j \in Y} t_1[A_j] \approx_{\lambda[A_j]} t_2[A_j]$, where $\lambda[A_i]$ and $\lambda[A_j]$ are the *matching similarity thresholds* on the attributes of A_i and A_j respectively. In the above constraint, for each attribute $A_i \in X \cup Y$, the similarity matching operator $\approx$ indicates true, if the similarity between $t_1[A_i]$ and $t_2[A_i]$ satisfies the corresponding threshold $\lambda[A_i]$. For example, a MD $\varphi([\mathsf{Street}] \rightarrow [\mathsf{City}], < 0.8, 0.7 >)$ in the Contacts relation denotes that if two tuples has similar Street (with matching similarity greater than 0.8) then their City values are probably similar as well (with similarity at least 0.7).

Like FDs and CFDs [17, 7], we adopt *support* and *confidence* measures to evaluate the matching dependencies. According to the above constraint of MDs, we need to consider the matching quality (e.g., cosine similarity or edit distance) of any pair of tuples t_1 and t_2 for $\mathcal{R}$. Therefore, we compute a statistical distribution (denoted by $\mathcal{D}$) of the 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 $\mathcal{D}$ corresponds to the matching quality values on the attribute A_i of $\mathcal{R}$, and P is the statistical value. Let s be a statistical tuple in $\mathcal{D}$. The statistic $s[P]$ denotes the probability that any two tuples t_1 and t_2 of $\mathcal{R}$ have the matching quality 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 $count(s)$ records the pairs of tuples having matching quality s. Different matching operators have various spaces of matching values, such as cosine similarity in $[0.0, 1.0]$ while edit distance having edit operations $1, 2, \ldots$. In order to evaluate in a consistent environment, we map these matching quality values $s[A]$ to a unified space, say $[0, d-1]$, which is represented by $\mathsf{dom}(A)$ with d elements. Table 3 shows an example of the statistical distribution $\mathcal{D}$ computed from Contacts in Table 1 by mapping[2] the cosine similarities in $[0.0, 1.0]$ to elements in $[0, d-1]$ of $\mathsf{dom}(A)$ with $d = 10$. According to $\mathsf{dom}(A)$ in our example, the first tuple $(1, 0, 3, \ldots, 0.065)$ denotes that there are about 6.5% matching pairs in all pair-wised tuple matching, whose similarities are $1, 0, 3, \ldots$ on the attribute $A_1, A_2, A_3, \ldots$ respectively.

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

A_1	A_2	A_3	A_4	A_5	A_6	P	
1	0	3	5	8	4	0.065	s_1
7	4	0	0	4	1	0.043	s_2
0	4	8	1	6	2	0.124	s_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

[1] The MDs syntax is described with two relation schema R_1, R_2 for object identification in [12], which can also be represented in a single relation schema R as the FDs.

[2] E.g., cosine similarity value s times $d-1$

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 matching similarity threshold pattern λ on the attributes of X and Y respectively in a MD φ, which are also specified in terms of elements in $\mathsf{dom}(A)$ 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 $\varphi(X \to Y, \lambda)$ are presented as follows:

$$\begin{aligned} \mathsf{support}(\varphi) &= P(X \vDash \lambda_X, Y \vDash \lambda_Y) \\ &= \sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z) \\ \mathsf{confidence}(\varphi) &= P(X \vDash \lambda_X \mid Y \vDash \lambda_Y) \\ &= \frac{\sum_Z P(X \vDash \lambda_X, Y \vDash \lambda_Y, Z)}{\sum_{Y,Z} P(X \vDash \lambda_X, Y, Z)} \end{aligned}$$

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.

Consider any two tuples t_1 and t_2 from the original data relation $\mathcal{R}$, the $\mathsf{support}(\varphi)$ 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 $\mathsf{confidence}(\varphi)$ 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 $\mathsf{confidence}(\varphi)$ 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.

3.2 Extended Matching Dependencies

Given a database instance, the expressive power of MDs may not strong enough to capture the minimum requirement η_s and η_c of support and confidence respectively. For example, as the Contacts example shown in Table 1, Claire Green have high similarities (always greater than 0.9) to the name of the same person with typo such as Claire Greem. We have 3 pairs of tuples to support a MD $\varphi([\mathsf{Name}] \to [\mathsf{SIN}], < 0.9, 1.0 >)$ with 100% confidence according to tuples in Contacts table. On the other hand, we can observe that the matching similarity of Jason Smith and its abbreviation format J.Smith is low (say 0.6), but these two tuples denote the same person in the real world with a matched SIN. However, J.Smith and W.J.Smith are two persons with different SIN, their similarity equals to 0.85 which is higher than the above 0.6. Assume that users require that a MD having $\eta_c = 1.0$ and $\eta_s = \frac{4}{15}$, where 15 is the total number of matching pairs of original 6 tuples according to the example in Table 1. If the matching similarity threshold on attribute Name is high (≥ 0.9), then the support is $\frac{3}{15}$ which is lower than η_s. On the other hand, simply lowering down the similarity threshold on Name attribute from 0.9 to 0.6 will increase the support, but the corresponding confidence is decreased to $\frac{4}{5} < \eta_c$ due to the counting of false case (J.Smith and W.J.Smith). In other words, we might not be able to capture this dependency with desired support and confidence using single similarity thresholds in MDs. Thus, for the above example case, instead of using single similarity threshold like $\mathsf{Name} \geq 0.6$, we introduce the intervals of matching similarities on attribute Name, e.g., $\mathsf{Name} \vDash \{[0.6, 0.8), [0.9, 1.0]\}$, to exclude the false cases. Therefore, by using more expressive similarity intervals, we can find $[\mathsf{Name}] \to [\mathsf{SIN}]$ with required support and confidence.

Formally, we define eMDs over a relation instance of $\mathcal{R}$ by:

DEFINITION 2. *An* extended matching dependency (eMD) ψ *is a pair* $(X \to Y, \mathcal{T})$*, where* $X \to Y$ *is the same as a standard* MD*; and* $\mathcal{T}$ *is a tableau of interval patterns with attributes of* $X \cup Y$*. Let* A *be an attribute in* $\mathcal{T}$ *and let* δ *be a* interval pattern *in* $\mathcal{T}$*. The value of* $\delta[A]$ *is a* matching similarity interval*, denoted by* $[v, u)$*, specifying a set of matching similarity values from* v *to* $u - 1$ *in* $\mathsf{dom}(A)$*, where* $v, u \in \mathsf{dom}(A)$*,* $1 \leq v < u \leq d$*, and* d *is the size of* $\mathsf{dom}(A)$*.*

Table 4 shows an example of interval pattern tableau $\mathcal{T}$ for an eMD $([A_2, A_5, A_6] \to [A_1, A_4], \mathcal{T})$. The semantics of an interval pattern δ is similar to the similarity threshold pattern in MDs. For example, we consider the intervals specified by the first pattern in Table 4. For any pair of data tuples from the original relation $\mathcal{R}$, if their matching similarities are in $[8, 10)$ of attribute A_2, in $[1, 5)$ of attribute A_5, and $[7, 8)$ of attribute A_6, then they should match on A_1 and A_4 with similarities in $[8, 10)$ and $[9, 10)$, respectively.

Table 4: Example pattern tableau for eMD

A_2	A_5	A_6	A_1	A_4
$[8, 10)$	$[1, 5)$	$[7, 8)$	$[8, 10)$	$[9, 10)$
$[7, 8)$	$[5, 6)$	$[6, 8)$	$[8, 10)$	$[9, 10)$
$[3, 6)$	$[9, 10)$	$[8, 9)$	$[7, 9)$	$[8, 9)$

When we have $u = d$, the semantics of the matching similarity interval $\delta[A] = [v, u)$ in a pattern tableau $\mathcal{T}$ is exactly the same as the matching similarity threshold $\lambda[A] = v$ in the standard MDs. That is, both of them denote the set of $d - v$ values in $\mathsf{dom}(A)$ from v to $(d - 1)$. Therefore, as shown in Table 5, a standard MD, e.g., $([A_5, A_6] \to [A_1, A_2], < 6, 8, 9, 8 >)$ can also be represented by an eMD with pattern tableau.

Table 5: Example pattern tableau for representing MD

A_5	A_6	A_1	A_2
$[6, 10)$	$[8, 10)$	$[9, 10)$	$[8, 10)$

Next, we study the evaluation of eMDs with support and confidence based on the statistical distribution $\mathcal{D}$ as well. Given an eMD $\psi(X \to Y, \mathcal{T})$ and a statistical distribution $\mathcal{D}$, let s be any statistical tuple in $\mathcal{D}$ and δ be any interval pattern in $\mathcal{T}$. The *satisfiability* relationship $\vDash$ is defined as follows. Let δ_X and δ_Y be the projections of similarity interval pattern δ on attributes X and Y respectively. For all attributes $\forall A$ in X, if $s[A] = a$ is in the interval specified by $\delta[A] = [v, u)$, e.g., $2 \vDash [1, 3)$, we say that the similarity values $s[X]$ of s on attributes X can *satisfy* the similarity intervals specified by δ_X, denoted as $s[X] \vDash \delta_X$. We can also develop the same relationship on δ_Y.

Similar to the semantics of confidence for a MD, the confidence of each pattern δ in the eMD ψ can be computed by

$$\mathsf{confidence}(\delta) = P(Y \vDash \delta_Y \mid X \vDash \delta_X)$$

Often, users may expect that all the patterns in the eMD pattern tableau can be utilized with high confidence. Thus, we define the minimum pattern confidence as the eMD confidence.

$$\mathsf{confidence}(\psi) = \min_{\delta \in \mathcal{T}} \mathsf{confidence}(\delta)$$

The support is defined as the total proportion of matching pairs that satisfy the intervals of patterns in the tableau $\mathcal{T}$ of eMD.

$$\mathsf{support}(\psi) = P([X, Y] \vDash \mathcal{T})$$

where $[X, Y] \vDash \mathcal{T}$ denotes that the similarity values on X and Y can satisfy at least one pattern δ in $\mathcal{T}$. Both these two probabilities can be calculated based on the statistical distribution $\mathcal{D}$.

4. DISCOVERING MDs

We now study the determination of matching similarity threshold pattern for MDs based on the statistical distribution, which is a new problem different from FDs. In fact, once the $X \rightarrow Y$ is given for a FD, it already implies the similarity threshold to be 1.0, that is, $(X \rightarrow Y, < 1.0, 1.0 >)$ if it is represented by the MD syntax. Unlike FDs, we have various settings of matching similarity thresholds for MDs. Therefore, in this section, we discuss how to find the right similarity thresholds in order to discover the MDs satisfying the required support and confidence.

4.1 Threshold Determination Problem

In order to discover a 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 discovered MDs to guide objet identification in the Contacts table, then $Y = \mathsf{SIN}$. The thresholds λ_Y is often set to a high similarity threshold by applications to ensure well match on Y attributes. For example, λ_Y is set to 1.0 for $Y = \mathsf{SIN}$ in the object identification application. Note that without the preliminary λ_Y, the discovered MDs will be meaningless. For example, a MD with $\lambda_Y = 0$ can always satisfy any requirement of η_c, η_s. Since all the statistical tuples can satisfy the thresholds $\lambda_Y = 0$, the corresponding support and confidence will always be equal to 1.0.

DEFINITION 3. *The threshold determination problem of* MDs *is: given the minimum requirements of support and confidence η_s, η_c and the matching similarity threshold pattern λ_Y, to find all the* MDs *$\varphi(X \rightarrow Y, \lambda)$ with threshold pattern λ_X on attributes X having $\mathsf{confidence}(\varphi) \geq \eta_c$ and $\mathsf{support}(\varphi) \geq \eta_s$, if exist; otherwise to return* infeasible.

The attributes X can be initially assigned by $\mathcal{R} \setminus Y$ if no suggestion is provided by specific applications, since our discovery process can automatically remove those attributes that are not required in X for a MD φ. Specifically, when a possible discovered threshold $\lambda[A]$ on attribute A is $0 \in \mathsf{dom}(A)$, it means that any matching 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 of the MD φ.

4.2 Exact Algorithm

Now, we present an algorithm to compute the matching 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. For simplicity, we use λ to denote the threshold pattern project λ_X with $\lambda[A_1], \ldots, \lambda[A_{m_X}]$ on all the m_X attributes of X. Since, each threshold $\lambda[A]$ on attribute A is a value from $\mathsf{dom}(A)$, i.e., $\lambda[A] \in \mathsf{dom}(A)$, 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

$$\mathcal{C}_t = \mathsf{dom}(A_1) \times \cdots \times \mathsf{dom}(A_{m_X}) = \mathsf{dom}(X).$$

The total number of candidates is $c = |\mathcal{C}_t| = |\mathsf{dom}(X)| = d^m$, where d is the size of $\mathsf{dom}(A)$.

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 $\mathsf{support} = P_n^j \geq \eta_s$ and $\mathsf{confidence} = \frac{P_n^j(X,Y)}{P_n^j(X)} \geq \eta_c$.

Algorithm 1 Exact algorithm **EA**$(\mathcal{D}, \mathcal{C}_t)$

```
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, Y), P_i^j(X)
5: return 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, whose time complexity is $\mathcal{O}(nc)$.

4.3 Pruning Strategies

Since 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. In fact, 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 two subsections, we present pruning techniques based on the given support and confidence, respectively.

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 4. *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 *dominate* 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 $\mathsf{support}(\varphi_1) \geq \mathsf{support}(\varphi_2)$.*

According to Lemma 1, given a candidate similarity threshold pattern λ_j having support lower 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 have support lower than η_s and can be safely pruned without computing their associated support and confidence.

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 DAG (directed acyclic graph), $\mathcal{G}$, to represent candidate similarity patterns as vertices and dominant relationships among the similarity patterns as edges. Thus, the dominant order of candidate patterns can be obtained by a BFS traversal upon $\mathcal{G}$.

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

We first group the statistical tuples in $\mathcal{D}$ into two parts based on the preliminary λ_Y as follows. Let k be a pivot between 1 and n.

For the first k tuples, we have $s_i[Y] \vDash \lambda_Y, 1 \le i \le k$. All the remaining $n-k$ tuples have $s_i[Y] \nvDash \lambda_Y, k+1 \le i \le n$. This grouping of statistical tuples in $\mathcal{D}$ can be done in linear time.

LEMMA 2. *Consider a pre-grouped statistical distribution $\mathcal{D}$. For any $1 \le i_1 < i_2 \le n$, we always have $\frac{P^j_{i_1}(X,Y)}{P^j_{i_1}(X)} \ge \frac{P^j_{i_2}(X,Y)}{P^j_{i_2}(X)}$.*

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 $\mathcal{D}$.

Algorithm 2 Pruning by support & confidence **EPSC**$(\mathcal{D}, \mathcal{C}_t)$

```
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 tuple s_i ∈ D, i : 1 → n do
4:       compute P_i^j(X,Y), P_i^j(X)
5:       if P_i^j(X,Y)/P_i^j(X) < η_c then
6:          remove λ_j from C_t                {Pruning by confidence}
7:          if P_i^j(X,Y) ≥ η_s then
8:             break
9:    if P_n^j(X,Y) < η_s then
10:      remove all the remaining candidates λ' dominated by λ_j
         from C_t                              {Pruning by support, λ' ≽ λ_j}
11: return λ_j with confidence and support satisfying η_c, η_s
```

Finally, both the pruning by support and the pruning by confidence are cooperated together into a single threshold determination algorithm as shown in Algorithm 2(namely EPSC). We also demonstrate the performance of the hybrid pruning EPSC in Section 6.

4.4 Approximation Algorithm

Though we have proposed pruning rules for exact method (Algorithm 2), the whole evaluation space is still all the n tuples in statistical distribution $\mathcal{D}$. Therefore, in this section, we present an approximate algorithm which only traverses the first k ($k = 1, \dots n$) tuples in $\mathcal{D}$, with bounded relative errors on support and confidence of returned MDs.

Let C^n and S^n be the confidence and support computed in the exact solution with all n 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. For a candidate threshold pattern $\lambda_j \in \mathcal{C}_t$, let

$$\beta = P^j_k(X), \quad \bar{\beta} = P^j_n(X) - P^j_k(X)$$

where β denotes $P(X \vDash \lambda_X)$ for the candidate λ_j based on the first k tuples in $\mathcal{D}$, and $\bar{\beta}$ 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 $\bar{\beta}$ for a specific k is in a certain range.

LEMMA 3. *If we have $\bar{\beta} \le \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$, 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 SKETCH. Let

$$\alpha = P^j_k(X,Y), \quad \bar{\alpha} = P^j_n(X,Y) - P^j_k(X,Y)$$

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

$$1 + \frac{\bar{\beta}}{\beta} \ge Z = \frac{\alpha(\beta+\bar{\beta})}{\beta(\alpha+\bar{\alpha})} \ge \frac{\beta+\bar{\beta}}{\beta + \frac{\bar{\beta}}{\eta_c}} \tag{1}$$

Referring to the minimum support requirement, for a valid λ_j, we have $\beta \ge \alpha \ge \eta_s$. Moreover, since $\bar{\beta} \le \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$, we also have $\bar{\beta} \le \epsilon\eta_s$ and $\bar{\beta} \le \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}$. Therefore,

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

On the other hand, according to the computation of support, having $S^k = \alpha$ and $S^n = \alpha + \bar{\alpha}$, we can also prove

$$\frac{S^n - S^k}{S^n} = \frac{1}{1 + \frac{\alpha}{\bar{\alpha}}} \le \frac{1}{1 + \frac{1}{\epsilon}} < \epsilon$$

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

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

THEOREM 1. *Given an error bound $\epsilon, 0 < \epsilon < 1 - \eta_c$, we can determine a minimum k, having $\bar{B}(k) \le \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$, $1 \le k \le n$. The approximation by considering first k 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 $\mathcal{O}(kc)$.*

Finally, we present the approximation implementation in Algorithm 3. Let $\bar{B}$ denotes $\bar{B}(k) = \sum_{i=k+1}^{n} s_i[P]$ for the current k. With k decreasing from n to 1, we can determine a minimum k where $\bar{B} = \bar{B}(k) \le \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$ 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 $\bar{B} = \bar{B}(k) \le \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$, the processing terminates. Here, the error bound ϵ is specified by user requirement with $0 < \epsilon < 1 - \eta_c$.

Algorithm 3 Approximation algorithm **AP**$(\mathcal{D}, \mathcal{C}_t)$

```
1: for each tuple s_k ∈ D, k : n → 1 do
2:    B̄ += s_k[P]
3:    if B̄ > min(εη_s, εη_sη_c/(1−ε−η_c)) then
4:       k++; break                            {Compute k}
5: for each candidate λ_j ∈ C_t, j : 1 → c do
6:    P_0^j(X,Y) = P_0^j(X) = 0
7:    for each tuple s_i ∈ D, i : 1 → k do
8:       compute P_i^j(X,Y), P_i^j(X)
9: return λ_j with confidence and support satisfying η_c, η_s
```

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 $\bar{B}(k) = \sum_{j=k+1}^{n} s_j[P]$ 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 reorganize the tuples in $\mathcal{D}$ in the decreasing order

of P as the input of Algorithm 3. 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.

Approximation Individually. We study the approximation by each individual candidate λ_j with a more efficient bound condition respectively. According to formula (1) in the proof of error bound, we find that for each specific candidate λ_j if $\bar{\beta} \leq \min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c})$, 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 $\bar{\beta}$ can be determined dynamically for each candidate λ_j respectively during the processing. Algorithm 4 shows the implementation of approximation with dynamic bound condition on each candidate λ_j individually.

Algorithm 4 Approximation individually **API**($\mathcal{D}, \mathcal{C}_t$)

```
1: for each tuple s_i ∈ D, i : n → 1 do
2:    B̄ += s_i[P]
3:    if B̄ ≤ min(εη_s, εη_sη_c/(1−ε−η_c)) then
4:       k = i                                   {Compute k}
5: for each candidate λ_j ∈ C_t, j : 1 → c do
6:    P_0^j(X,Y) = P_0^j(X) = 0
7:    B̄_j = B̄
8:    for each tuple s_i ∈ D, i : 1 → k do
9:       compute P_i^j(X,Y), P_i^j(X)
10:      β = P_i^j(X)
11:      B̄_j −= s_i[P]
12:      if B̄_j ≤ min(εβ, εβη_c/(1−ε−η_c)) then
13:         break
14: return λ_j with confidence and support satisfying η_c, η_s
```

COROLLARY 1. *The worst case complexity of the approximation individually is* $\mathcal{O}(kc)$

PROOF. Note that with the increasing of i from 1 to k, for a specific λ_j, the value β increases and $\bar{B}_j$ decreases. For any $i < k$, if $\beta < \eta_s$, i.e., λ_j is invalid currently, the bound condition cannot be satisfied having $\min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c}) < \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}) < \bar{B}_j$. When λ_j has $\beta \geq \eta_s$ as a valid threshold, the bound condition is relaxed from $\min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c})$ to $\min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c})$. Thereby, the bound condition may be satisfied by a smaller i than k, i.e., $\min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}) < \bar{B}_j \leq \min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c})$. The worst case is that all candidates do not achieve their bounds until processing the tuple s_k, where $\bar{B}_j = \bar{B}(k) \leq \min(\epsilon\eta_s, \frac{\epsilon\eta_s\eta_c}{1-\epsilon-\eta_c}) \leq \min(\epsilon\beta, \frac{\epsilon\beta\eta_c}{1-\epsilon-\eta_c})$ must be satisfied. This is exact the Algorithm 3 without individual approximation. □

Finally, we cooperate the pruning by support together with the approximation (namely APS) and the approximation individually (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).

5. DISCOVERING eMDs

Rather than finding single matching similarity threshold for each attribute of X for MDs, the discovery of eMDs is even more complicated, where the pattern tableau specifies a set of patterns with various matching similarity intervals. Therefore, given a requirement of support and confidence, we have various interval pattern combinations to form a tableau. It is naturally desirable to discover the most concise pattern tableau for eMDs, that is, a minimum set of patterns that can still meet users' requirements. In this section, we first formalize the optimization problem of discovering eMDs, then we present the algorithms to find desired eMDs efficiently.

5.1 Optimal Tableau Problem

As mentioned, the patterns in a tableau may cover the same semantics of dependencies as redundancy. Real applications such as violation detection are often interested in concise eMDs with the minimum size of pattern rules in the pattern tableau $\mathcal{T}$. To discover such eMDs, again, the preliminary specifies what the *true* constraint of matching similarities is for attributes in Y, i.e., δ_Y. The optimization problem then targets to discover the optimal eMD ψ that can infer this true constraint of Y.

DEFINITION 5. *The problem of discovering the optimal e*MD *is: given the minimum requirements of support and confidence* η_s, η_c *and the matching similarity interval pattern* δ_Y*, to find the optimal e*MD $\psi(X \rightarrow Y, \mathcal{T})$ *that* minimizes *the size of the pattern tableau* $\mathcal{T}$ *with discovered patterns of intervals* δ_X *on attributes* X*, if one satisfies* η_s, η_c*; otherwise to return* infeasible.

For a certain δ_Y, to minimize the pattern tableau is equivalent to minimize the pattern sets with intervals on X. To be convenient, we use δ to denote the pattern projection δ_X on X. Since the distributions of X are independent given different Y values, if there are several different δ_Y of Y in $\mathcal{T}$, we can consider the minimization of each one individually. Thus, in the following work, we focus on the optimization with an individual δ_Y.

THEOREM 2. *The optimal e*MD *generation problem with minimum tableau size is* NP*-complete.*

PROOF SKETCH. We can show that the *3-partite graph vertex cover* problem [8], which is NP-complete, is polynomial-time reducible to the optimal eMD generation problem. □

5.2 Greedy Algorithm

Since discovering optimal eMDs is NP-complete, motivated by the greedy approximation for the *partial covering* problems [16, 17], we also study the greedy algorithm for discovering near optimal eMDs in polynomial time. The greedy algorithm we proposed below has two steps: **(I)** generating a set $\mathcal{C}_e$ of all the possible *interval pattern candidates* for tabular $\mathcal{T}$, and **(II)** selecting minimum number of patterns from $\mathcal{C}_e$ to satisfy the given support and confidence requirements. Before we illustrate the detailed steps of the algorithm, we first define some terms that will be used.

Consider a statistical tuple s in $\mathcal{D}$. For each attribute A, we define $\mathsf{int}(s[A]) = \{[v,u) \mid s[A] \vDash [v,u)\}$ to be all the similarity intervals $[v,u)$ that the value $s[A]$ can satisfy. For example, let $s[A] = \alpha \in \mathsf{dom}(A)$, then we have

$$\begin{aligned}\mathsf{int}(\alpha) &= \{[v,u) \mid 0 \leq v \leq \alpha, \alpha+1 \leq u \leq d\} \\ &= \{[0,\alpha+1), \ldots, [0,d), [1,\alpha+1), \ldots, [1,d), \ldots, \\ &\quad [\alpha,\alpha+1), \ldots, [\alpha,d)\}\end{aligned}$$

where d be the domain size of $\mathsf{dom}(A)$. The size of $\mathsf{int}(s[A])$ in worst case is $\mathcal{O}(d^2)$.

For the m_X attributes in X, let $\mathsf{pat}(s)$ be the set of all the patterns δ that s can satisfy, that is, the set of pattern candidates δ can be generated from s.

$$\mathsf{pat}(s) = \{\delta \mid s[X] \vDash \delta\} = \mathsf{int}(s[A_1]) \times \cdots \times \mathsf{int}(s[A_{m_X}])$$

Therefore, the size of $\mathsf{pat}(s)$ is $\mathcal{O}(d^{2m})$. Let c be the domain size of $\mathsf{dom}(X)$, having $c = d^m$. Thus, we have $|\mathsf{pat}(s)| = \mathcal{O}(c^2)$.

Given a pattern δ, let $\mathsf{cover}(\delta)$ record all the statistical tuples $s \in \mathcal{D}$ that satisfy δ, i.e.,

$$\mathsf{cover}(\delta) = \{s \mid s[X] \vDash \delta, s \in \mathcal{D}\}$$

Generation step. We first generate the candidate set $\mathcal{C}_e$ of all the possible patterns (namely PC in Algorithm 5) by considering the $\mathsf{pat}(s)$ of each statistical tuple s in $\mathcal{D}$.

Algorithm 5 Pattern candidates **PC**($\mathcal{D}$)

```
1: C_e = ∅
2: for each statistical tuple s ∈ D do
3:    for each pattern candidate δ ∈ pat(s) do
4:       if δ ∈ C_e then
5:          insert s to cover(δ)
6:          update confidence and support of δ to C_e
7:       else
8:          cover(δ) = {s}
9:          compute confidence and support of δ, insert δ to C_e
10: return C_e
```

Elimination step. After we have the pattern candidate set $\mathcal{C}_e$, the next step is to generate a minimum set of patterns from $\mathcal{C}_e$ as $\mathcal{T}$ of the eMD. Specifically, the greedy algorithm removes a pattern candidate δ with maximum support from $\mathcal{C}_e$ in each iteration, adds it into $\mathcal{T}$ if valid, and does not stop until the minimum support η_s is satisfied or all the valid pattern candidates are added to $\mathcal{T}$. Given a statistical tuple s, there may exist many patterns δ being satisfied by s. However, when we compute the support and confidence for each eMD, s is only assigned to one pattern and count towards its support. To follow this principle, we eliminate all the statistical tuples in $\mathsf{cover}(\delta)$ from the remaining $\delta' \in \mathcal{C}_e$, i.e., $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ in Algorithm 6.

Algorithm 6 Greedy algorithm **GA**($\mathcal{C}_e$)

```
1: T = ∅
2: support(ψ) = 0
3: while C_e ≠ ∅ and support(ψ) < η_s do
4:    δ = arg max_{δ∈C_e} support(δ)
5:    remove δ from C_e
6:    if confidence(δ) ≥ η_c then
7:       insert δ to T
8:       support(ψ) += support(δ)
9:       for each pattern candidate δ' ∈ C_e do
10:         cover(δ') = cover(δ') \ cover(δ)
11:         update confidence and support of δ' to C_e
12: return T
```

THEOREM 3. *The greedy algorithm finds an approximately optimal* eMD *with an error bound on the tableau size $|\mathcal{T}|$ compared to an optimal* eMD *tableau size $|\mathcal{T}^*|$, having $|\mathcal{T}|/|\mathcal{T}^*| = \ln n + 1$. The complexity is $\mathcal{O}(nc^2)$.*

5.3 Pruning Strategies

As shown in Theorem 3, it is still very costly to compute the pattern tabular for an eMD. This is because both generation and elimination steps require scanning all the possible pattern candidates. In fact, not all patterns that s satisfies should be generated as Algorithm 5 does. Moreover, in the elimination step, after moving a pattern to $\mathcal{T}$, it is not necessary as well to update supports and confidences of all the rest pattern candidates in $\mathcal{C}_e$. Therefore, in the rest of this section, we propose pruning strategies for generation and elimination steps, respectively.

Pruning during generation. We first study the relationship among patterns, based on which, we can find out redundant pattern candidates during the generation.

DEFINITION 6. *For any two intervals $[v, u)$ and $[g, h)$, if $v \leq g$ and $h \leq u$, then $[v, u)$* dominates *$[g, h)$, denoted by $[v, u) \lessdot [g, h)$.*

Then, the relationships among the intervals on an attribute can be represented by a directed acyclic graph. For example, as shown in Figure 1 (a), each black node denotes a possible interval. An arrow from node a to b denotes $a \lessdot b$. For each attribute A, there is a triangle structure that specifies all the possible intervals corresponding to this attribute, e.g., Figure 1 (b) for attribute A_2, Figure 1 (c) for attribute A_3, etc. Each pattern δ, thereby, consists of exact one node (interval) from each triangle (attribute).

DEFINITION 7. *Consider any two patterns δ_1 and δ_2. For all attribute $\forall A \in X$, if the intervals satisfy $\delta_1[A] \lessdot \delta_2[A]$, then we say δ_1 dominates δ_2, denoted by $\delta_1 \lessdot \delta_2$.*

If there exists an attribute $\exists A \in X$, having intervals $\delta_1[A] \lessdot \delta_2[A]$, then we say δ_1 partially *dominates δ_2, denoted by $\delta_1 \lessdot^p \delta_2$.*

LEMMA 4. *For any two patterns δ_1 and δ_2, if δ_1 dominates δ_2, i.e., $\delta_1 \lessdot \delta_2$, then we have $\mathsf{cover}(\delta_2) \subseteq \mathsf{cover}(\delta_1)$ and $\mathsf{support}(\delta_1) \geq \mathsf{support}(\delta_2)$. When $\mathsf{cover}(\delta_1) = \mathsf{cover}(\delta_2)$, we say that patterns δ_1 and δ_2 are equivalent, having $\mathsf{support}(\delta_1) = \mathsf{support}(\delta_2)$.*

Now, we study the pruning technique for generating pattern candidates as less as possible. Note that some of the instances (say α) in $\mathsf{dom}(A)$ of an attribute A may not appear in a certain distribution $\mathcal{D}$. However, according to the pattern candidate generation algorithm, these instances $\alpha \in \mathsf{dom}(A)$ are still considered as the bounds of intervals in candidate patterns. We first study the pruning strategies based on these non-appearing instances.

Intuitively, since the value α does not appear in attribute A in $\mathcal{D}$, all the patterns δ containing the interval $\delta[A] = [\alpha, \alpha + 1)$ on A should have an empty cover set and can be ignored directly. Moreover, consider some other patterns δ with intervals like $\delta[A] = [\alpha, \alpha + 2)$ on A. We can prove that there always exists another pattern δ' (such as $\delta'[A] = [\alpha+1, \alpha+2)$ on A) having $\mathsf{cover}(\delta) = \mathsf{cover}(\delta')$. According to Lemma 4, the pattern δ is equivalent to δ' and can be pruned as well. We formally define these candidate patterns with certain pruning intervals as follows.

THEOREM 4. *Consider any value $\alpha \in \mathsf{dom}(A)$. If this value α does not appear in the attribute A in $\mathcal{D}$, then all the pattern candidates that contain the following intervals on attribute A can be pruned in the candidate set $\mathcal{C}_e$: $I_1 = \{[\alpha, \alpha + u) \mid u = 1, 2, \dots\}$ and $I_2 = \{[\alpha - u + 1, \alpha + 1) \mid u = 1, 2, \dots\}$.*

PROOF SKETCH. For any pattern δ having $\delta[A] \in I_1$, we can always find a δ_2, having $\delta_2[A] = [\alpha+1, \alpha+u)$, which is equivalent to δ, i.e., $\mathsf{cover}(\delta) = \mathsf{cover}(\delta_2)$. Thus, the pattern δ can be pruned as redundancy. Similarly, for any pattern δ having $\delta[A] \in I_2$, we can also find a δ_1, having $\delta_1[A] = [\alpha-u+1, \alpha)$, as the redundancy of δ. □

For example, in Figure 1 (a), suppose that the item 4 does not appear in attribute A_1 in $\mathcal{D}$. Then, the sets of intervals, $I_1 = \{[4, 5), [4, 6), [4, 7), [4, 8), [4, 9), [4, 10)\}$ and $I_2 = \{[4, 5), [3, 5), [2, 5), [1, 5), [0, 5)\}$ marked by shade area in Figure 1 (a), can be ignored in attribute A_1 during the candidate generation. In other words, the pattern candidate generation with pruning (namely PCP) removes all the patterns with intervals from I_1 or I_2 in A_1 from $\mathcal{C}_e$.

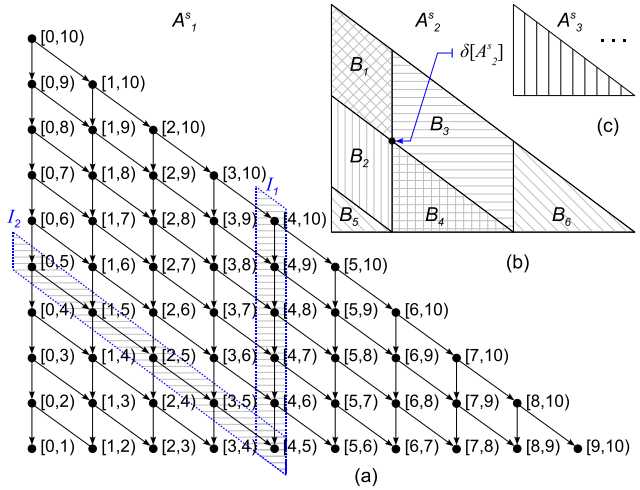

Figure 1: Domination relationship among intervals

Pruning during elimination. Next, we study the pruning of candidate patterns during the greedy computation. The major cost of elimination step ordinates from the updating $\mathsf{cover}(\delta')$ after moving the pattern δ with highest support to $\mathcal{T}$. Thus, we propose pruning rules to reduce the number of updates and remove the redundant patterns based on the *dominate* relationship among patterns.

Let δ be the current pattern in the greedy algorithm. Let $[v, u)$ denote the interval of δ on attribute A, i.e., $\delta[A] = [v, u)$. In order to develop the pruning technique, for each attribute A, we group all the intervals into 6 blocks according to the domination relationship on $[v, u)$ as follows.

$$
\begin{aligned}
B_1[A] &= \{[g,h) \mid [g,h) \lessdot [v,u)\}\\
B_2[A] &= \{[g,h) \mid [0,v+u) \lessdot [g,h) \lessdot [v,v+1)\}\\
B_3[A] &= \{[g,h) \mid [v,d) \lessdot [g,h) \lessdot [v+u-1,v+u)\}\\
B_4[A] &= \{[g,h) \mid [v,u) \lessdot [g,h)\}\\
B_5[A] &= \{[g,h) \mid [0,v) \lessdot [g,h)\}\\
B_6[A] &= \{[g,h) \mid [v+u,d) \lessdot [g,h)\}
\end{aligned}
$$

Among the 6 blocks defined above, $B_1[A]$ represents all the intervals on A that dominate $\delta[A]$, $B_4[A]$ denotes all the intervals on A that are dominated by $\delta[A]$, $B_5[A]$ and $B_6[A]$ are the intervals that have no overlapping with $\delta[A]$, and $B_2[A]$ and $B_3[A]$ are the intervals that have overlapping with $\delta[A]$. For example, in Figure 1 (b), we illustrate the 6 blocks of all the intervals in attribute A_2 based on the interval $\delta[A_2]$ of the current δ.

Based on these six partitioned blocks of intervals, we first identify the set of patterns δ' that are not updated by the elimination operation $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ even in the original greedy algorithm (Algorithm 6).

LEMMA 5. *Consider the current δ with the maximum support in $\mathcal{C}_e$. After inserting δ as a pattern in $\mathcal{T}$, the following sets of candidate patterns δ' are* not *updated:*

$$
\begin{aligned}
C_5 &= \{\delta' \mid \exists A, \delta'[A] \in B_5[A]\}\\
C_6 &= \{\delta' \mid \exists A, \delta'[A] \in B_6[A]\}
\end{aligned}
$$

PROOF SKETCH. For any pattern $\delta' \in C_5$ or $\delta' \in C_6$, we can prove that the intersection of cover sets of δ and δ' is $\mathsf{cover}(\delta') \cap \mathsf{cover}(\delta) = \emptyset$. In other words, the operation $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ takes no effect on pattern δ'. Thus δ' is not updated. □

According to Lemma 5, all the patterns with intervals from B_5 or B_6 on any attribute will not be updated in the current iteration. In other words, only the patterns with all the intervals from B_1, B_2, B_3, B_4 will be updated, i.e., $\mathcal{C}_e \setminus (C_5 \cup C_6)$.

Now, we study the pruning rules for $\delta' \in \mathcal{C}_e \setminus (C_5 \cup C_6)$ to avoid updates. Based on the dominate relationship, we propose to filter out the following two types of patterns: **(I)** those patterns pattern δ' having $\mathsf{cover}(\delta') = \emptyset$ after the $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ operation. Thus, these patterns can be pruned without conducting the updating operation. **(II)** those patterns δ' that always have another pattern δ_1 in C_5 or C_6 having $\mathsf{cover}(\delta') = \mathsf{cover}(\delta_1)$, i.e., equivalent, after the $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ operation. Since this pattern δ_1 is reserved in $\mathcal{C}_e$ without updating in the current iteration, the equivalent one δ' can be pruned as redundancy.

Formally, we define the patterns that can be directly pruned from the candidate set $\mathcal{C}_e$ as follows.

THEOREM 5. *Consider the current δ with the maximum support in $\mathcal{C}_e$. After inserting δ as a pattern in $\mathcal{T}$, the following set of candidate patterns C_p can be pruned from $\mathcal{C}_e$:*

$$C_p = \{\delta' \mid \forall A, \delta'[A] \in (B_2[A] \cup B_3[A] \cup B_4[A])\}$$

PROOF SKETCH. For a pattern $\delta' \in C_p$, the interval $\delta'[A]$ of any attribute A comes either from $B_2[A]$, $B_3[A]$ or $B_4[A]$. Let

$$
\begin{aligned}
C_2 &= \{\delta' \mid \exists A, \delta'[A] \in B_2[A]\}\\
C_3 &= \{\delta' \mid \exists A, \delta'[A] \in B_3[A]\}\\
C_4 &= \{\delta' \mid \forall A, \delta'[A] \in B_4[A]\}
\end{aligned}
$$

having $C_p = C_2 \cup C_3 \cup C_4$.

For any pattern $\delta' \in C_2$ or $\delta' \in C_3$, we can prove that there always exists a pattern in the remaining candidate pattern sets (say $\delta_1 \in C_5$ or $\delta_2 \in C_6$) which is equivalent to δ' after the current elimination step. Thus, the pattern δ' can be pruned as duplicates.

For any pattern $\delta' \in C_4$, we can prove that $\mathsf{cover}(\delta') = \emptyset$ after the current elimination step, thereby δ' can be pruned. □

According to the above definition of C_p, a pattern $\delta' \in C_p$ only contains intervals from B_2, B_3, B_4 on all the attribute A. In other words, each $\delta' \in C_p$ does not contain any interval from B_1, B_5, B_6 on all the attributes. Let

$$C_1 = \{\delta' \mid \exists A, \delta'[A] \in B_1[A]\}$$

Then, we can also represent C_p by $C_p = \mathcal{C}_e \setminus (C_1 \cup C_5 \cup C_6)$.

Now, for each attribute A, let $\delta_5[A] = [0, v)$ and $\delta_6[A] = [v + u, d)$, where $\delta[A] = [v, u)$. According to the partial domination $\lessdot^p$ in Definition 7, we can rewrite $C_5 = \{\delta' \mid \delta_5 \lessdot^p \delta'\}$ and $C_6 = \{\delta' \mid \delta_6 \lessdot^p \delta'\}$. Moreover, the set of patterns C_1 can also be rewritten by $\{\delta' \mid \delta' \lessdot^p \delta\}$. Consequently, the pruned candidate set $C_p = \mathcal{C}_e \setminus (C_1 \cup C_5 \cup C_6)$ can be specified by

$$C_p = \{\delta' \mid (\delta' \lessdot^p \delta \textbf{ or } \delta_5 \lessdot^p \delta' \textbf{ or } \delta_6 \lessdot^p \delta') = \textbf{false}\}$$

For example, suppose $\delta[A_1] = [2, 7)$ in Figure 1 (a), then we have $\delta_5[A_1] = [0, 2), \delta_6[A_1] = [7, 10)$. We can also compute the intervals of δ_5, δ_6 on the other attributes A. According to Theorem 5, those remaining patterns can be safely pruned if they **cannot** satisfy the above partial domination relationships of δ_5, δ_6 and δ.

Finally, we present the greedy algorithm with pruning (namely GAP) in Algorithm 7. Note that calculating the boundary patterns δ_5, δ_6 is in constant time and the pruning can be applied recursively in the next elimination iteration.

In Algorithm 7, rather than removing each statistical tuple from possible patterns exactly once in the original greedy algorithm, we prune the patterns which are not necessary to conduct the operation $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$. Let $\gamma (0 \leq \gamma \leq 1)$ be the pruning rate on average, that is, γ percentage of candidate patterns can be avoided to perform the $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ operation. Then, the complexity of Algorithm 7 is $\mathcal{O}((1-\gamma)nc^2)$.

Algorithm 7 Greedy algorithm pruning **GAP**($\mathcal{C}_e$)

$\mathcal{T} = \emptyset$
$\mathsf{support}(\psi) = 0$
while $\mathcal{C}_e \neq \emptyset$ and $\mathsf{support}(\psi) < \eta_s$ **do**
  $\delta = \arg\max_{\delta \in \mathcal{C}_e} \mathsf{support}(\delta)$
  remove δ from $\mathcal{C}_e$
  if $\mathsf{confidence}(\delta) \geq \eta_c$ **then**
    insert δ to $\mathcal{T}$
    $\mathsf{support}(\psi)$ += $\mathsf{support}(\delta)$
    calculate δ_5, δ_6 from δ
    for each pattern candidate $\delta' \in \mathcal{C}_e$ **do**
      if $(\delta' \lessdot^p \delta$ **or** $\delta_5 \lessdot^p \delta'$ **or** $\delta_6 \lessdot^p \delta') =$ **false then**
        remove δ' from $\mathcal{C}_e$
      else
        $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$
        update confidence and support of δ' to $\mathcal{C}_e$
return $\mathcal{T}$

As illustrated in the experiments, the GAP can always improve the discovering efficiency. In fact, we can develop a minimum bound of the pruning rate γ on average as follows.

COROLLARY 2. *The pruning rate on average γ has a minimum bound $\gamma \geq 0.754^m$, where m is the number of attributes in X.*

PROOF. First, we consider the intervals in one single attribute A. According to Lemma 5, the patterns with intervals from B_5 or B_6 will not be affected in the current iteration. Therefore, the original greedy algorithm only updates the patterns with all the intervals from B_1, B_2, B_3 and B_4. Let $\delta[A] = [v, u)$ for the current pattern δ on attribute A. The total number of intervals in B_1, B_2, B_3, B_4 will be $\frac{1}{2}(d(d+1) - v(v+1) - (d-v-u)(d-v-u+1))$. Next, for the greedy algorithm with pruning, as we proved in Theorem 5, we can avoiding the update of patterns with all the intervals in B_2, B_3 and B_4, and remove them directly. Thus, we observe the number of intervals in B_1, i.e., $\frac{1}{2}((d-u)(d-u+1) - v(v+1) - (d-v-u)(d-v-u+1))$. Then, we consider the percentage p of intervals in B_1 compared to B_1, B_2, B_3, B_4, i.e., $p = \frac{(d-u)(d-u+1)-v(v+1)-(d-v-u)(d-v-u+1)}{d(d+1)-v(v+1)-(d-v-u)(d-v-u+1)}$ for each possible v, u. Let $x = d - u, y = v$, then we have the percentage p on average as follows.

$$p = \frac{1}{d}\sum_{x=0}^{d-1}\left(\frac{\sum_{y=0}^{x}\frac{x(x+1)-y(y+1)-(x-y)(x-y+1)}{d(d+1)-y(y+1)-(x-y)(x-y+1)}}{x+1}\right)$$
$$= \frac{1}{d}\sum_{x=0}^{d-1}\left(\frac{x+1-\sum_{y=0}^{x}\frac{d^2-x^2+d-x}{d^2-x^2+d-x+2y(x-y)}}{x+1}\right)$$

Note that we have $y(x-y) \leq x^2/4$ for $y \in [0, x]$, thus

$$p \leq \frac{1}{d}\sum_{x=0}^{d-1}\left(1 - \frac{d^2-x^2+d-x}{d^2-x^2/2+d-x}\right) \leq \frac{1}{d}\sum_{x=1}^{d}\frac{x^2}{2d^2-x^2}$$
$$= \frac{1}{\sqrt{2}}\sum_{x=1}^{d}\left(\frac{1}{\sqrt{2}d-x} + \frac{1}{\sqrt{2}d+x}\right) - 1$$

Since the harmonic number $H(n) = \sum_{k=1}^{n}\frac{1}{k} = \ln n + \mathcal{O}(1)$,

$$p \leq \frac{1}{\sqrt{2}}\big(H(\sqrt{2}d) - H(\sqrt{2}d-d) + H(\sqrt{2}d+d) - H(\sqrt{2}d)\big) - 1$$
$$= \frac{1}{\sqrt{2}}\ln\frac{\sqrt{2}+1}{\sqrt{2}-1} - 1 = 0.246$$

Next, we consider all the m attributes in δ. According to Theorem 5, the pruned patterns should have intervals from B_2, B_3, B_4 for all the m attributes. Thus, the pruning rate $\gamma = (1-p)^m \geq 0.754^m$. Finally, we prove that γ is bounded by 0.754^m. □

6. EXPERIMENTAL EVALUATION

Data sets. The *Cora*[3] data set, prepared by McCallum et al. [20], consists of 12 attributes including author, volume, title, institution, venue, etc. The *CiteSeer*[4] data set is selected with attributes including title, author, address, affiliation, subject, description, etc. We use the *cosine* similarity to evaluate the matching quality of the tuples in the original data. By applying the $\mathsf{dom}(A)$ mapping in Section 3, we can obtain statistical distributions with at most $186,031$ statistical tuples in *Cora* and $314,382$ statistical tuples in *CiteSeer*. Our experimental evaluation is then conducted in several pre-processed statistical distributions with various sizes of statistical tuples n from $10,000$ to $150,000$ respectively.

We mainly observes the efficiency of proposed algorithms. Since our main task is to discover MDs and eMDs 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 matching similarity settings of attributes for MDs and eMDs. Suppose that users want to discover the matching similarity settings for the dependencies author, volume, title $\rightarrow$ venue with the preliminary requirement of minimum similarity 0.6 on venue in *Cora*, and address, affiliation, description $\rightarrow$ subject with preliminary 0.1 on subject in *CiteSeer* respectively. A returned result is either infeasible, or a threshold pattern for MD or a tableau of interval patterns for eMD.

All the algorithms are implemented by Java. The experiment evaluates on a machine with Intel Core 2 CPU (2.13 GHz) and 2 GB of memory. The programs run entirely in main memory.

6.1 Evaluation on MDs

First, we evaluate the performance of pruning by support (EPS) compared with the original exact algorithm (EA). As shown in (a) and (b) in Figure 2 and 3, the EA, which verifies all the possible candidates, should have the same cost no matter how η_s and η_c set. The EPS achieves low time cost in all the statistical distributions, which is only about $1/10$ of that of the EA. To observe more accurately, we also plot the EPS time cost in Figure 2 (c) and (d) with the same settings respectively. According to the pruning strategy, the EPS performance is only affected by support requirement η_s. It is natural that a higher η_s turns to the better pruning performance. Therefore, EPS with $\eta_s = 0.04$ in Figure 2 (c) shows lower time cost, e.g., about 0.4s for 150k, than that of $\eta_s = 0.01$ in (d), e.g., 0.6s for the same 150k. Similar results with different η_s are also observed on *Cora*, which are not presented due to the limit of space.

More pruning and approximation results are reported in (c) and (d) in Figure 2 and 3, including the pruning by both support and confidence (EPSC), the approximation together with pruning by support (APS), and the approximation individually together with pruning by support (APSI). When the confidence requirement η_c is high, e.g., in Figure 3 (d), the EPSC can remove those low confidence candidates and shows 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 Figure 3 (c). Thus, the approximation approaches have lower time cost, especially the APSI. According to the definition of the bound condition of approximation approaches, not only the ϵ, but also η_s

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

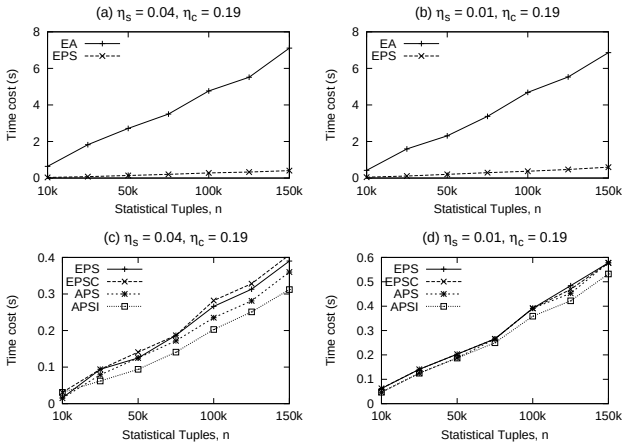

Figure 2: Approaches for MDs on *CiteSeer*

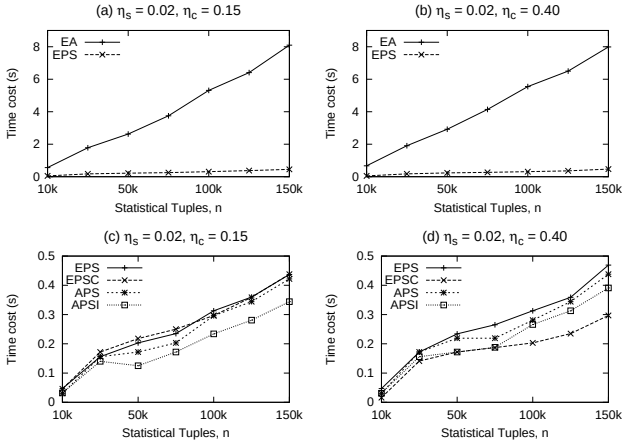

Figure 3: Approaches for MDs on *Cora*

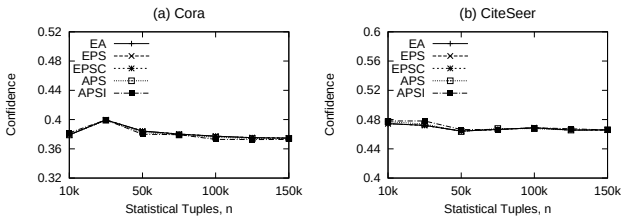

Figure 4: Approximate confidence of MDs

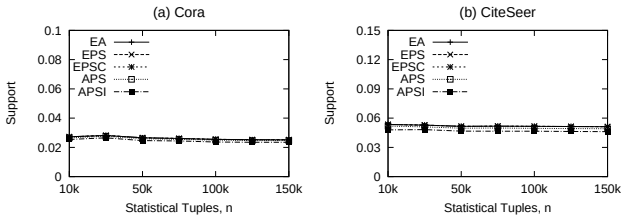

Figure 5: Approximate support of MDs

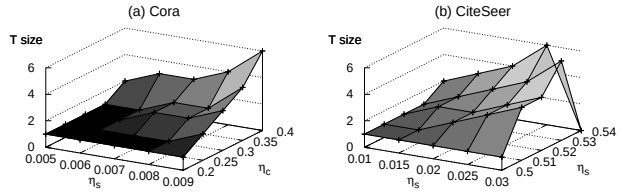

Figure 6: Various η_s and η_c for eMDs

affects the performance. As presented in Figure 2 (c), a higher η_s contributes a larger bound condition, which means the early termination of the program. Thus, approximation approaches show better performance in Figure 2 (c) compared with Figure 2 (d).

Finally, we evaluate the approximate confidence and support of the returned MDs with $\epsilon = 0.8$ on both two datasets in Figure 4 and 5. As we proved in Lemma 3, the error introduced in approximation approaches is bounded by ϵ on both confidence and support. Therefore, in Figure 4 and 5, the approximate confidence and support of APS and APSI are very close to those of exact algorithms.

6.2 Evaluation on eMDs

We first evaluate the influence of various η_s and η_c settings on the optimal tableau sizes in Figure 6. With the increase of the minimum support requirement η_s, we need to add more patterns into the tableau $\mathcal{T}$ of eMDs, and thus the $\mathcal{T}$ size increases as well. When the minimum confidence η_c is high at the same time, there might be not enough patterns to add into $\mathcal{T}$ that can achieve the η_s requirement. Therefore, the returned result will be infeasible ($\mathcal{T}$ size is 0), for example, in Figure 6 (b) with $\eta_s = 0.03$ and $\eta_c = 0.54$.

Next, we study the time performance of proposed algorithms, including the original pattern candidate generation (PC), the pattern candidate generation with pruning (PCP), the original greedy algorithm (GA), and the greedy algorithm with pruning (GAP). As shown in Figure 7 and 8, both the PCP and GAP techniques can improve the time performance compared with the original PC and GA respectively. Especially, the GAP works well (together with either PC or PCP) and keeps low time cost even when the GA requires about 30 times larger cost in the same environment. These results also verify our conclusion in Corollary 2 of the pruning rate bound on average. Moreover, the PCP is pruning based on the values that do not appear in the current distribution. When all the possible values appear, e.g., the 150k distribution of *CiteSeer* in Figure 8, the PCP does not work (either together with GA or GAP). Nevertheless, the PCP+GAP approach always achieve the best time performance.

The time performance of these greedy algorithm based techniques is also affected by different η_s and η_c requirements. When both the support and confidence requirements are high, e.g., $\eta_s = 0.007$ and $\eta_c = 0.4$ in Figure 7 (b), the algorithm needs to seek a large number of candidate patterns in order to satisfy the η_s and η_c requirements. The corresponding $\mathcal{T}$ sizes are large, as shown in (d). Consequently, the GA has high time cost when both η_s and η_c are high in (b). Next, if the minimum support is smaller, e.g., $\eta_s = 0.005$ in (a), the GA can terminate early, and thus has lower time cost. On the other hand, if the minimum confidence is smaller, e.g., $\eta_c = 0.2$ in (c), there are more candidate patterns that can be considered, some of which may have high supports. Since the GA greedily select the candidates with the largest support into $\mathcal{T}$ in each iteration, the number of patterns in $\mathcal{T}$ of $\eta_c = 0.2$ should not be larger than a higher $\eta_c = 0.4$. In fact, as shown in (d), the $\mathcal{T}$ size of $\eta_c = 0.2$ is quite small in all distributions, and the corresponding time cost in (c) is much lower than that of (b). Finally, although the GAP is affect by η_s and η_c as well, it can achieve significantly lower time cost than GA by avoiding unnecessary elimination operations.

We can observe similar conclusions in Figure 8 and also one more interesting result, i.e., the infeasible case. The answer in 120k is *infeasible* with $\mathcal{T}$ size 0 in Figure 8 (d) under the high $\eta_s = 0.01$ and $\eta_c = 0.55$ requirement. Recall that an *infeasible* answer is returned when the algorithm cannot achieve η_s after checking and

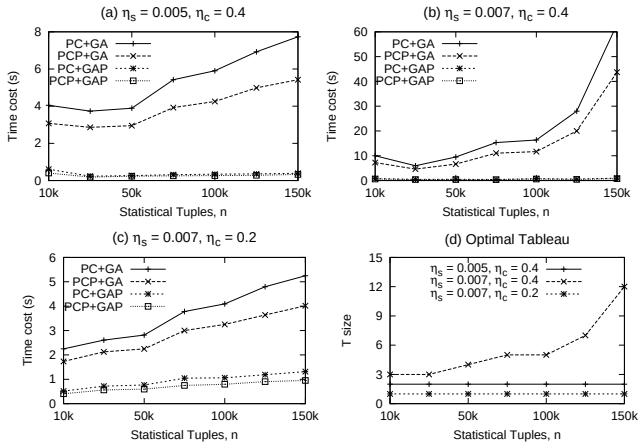

Figure 7: Approaches for eMDs on *Cora*

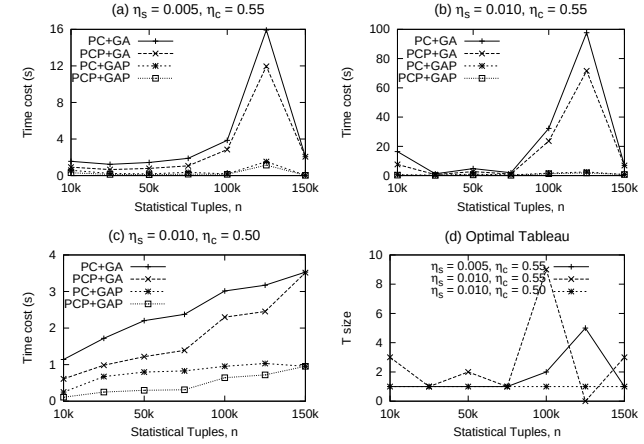

Figure 8: Approaches for eMDs on *CiteSeer*

adding all possible candidate patterns into $\mathcal{T}$. In other words, it is the worst case to traverse all the candidates when an *infeasible* answer returns. Therefore, as shown in Figure 8 (b), the corresponding time cost of 120k is the highest. Finally, due to the greedy strategy, the $\mathcal{T}$ sizes are various under different statistical distributions, e.g., for $\eta_s = 0.005, \eta_c = 0.55$ in Figure 8 (d), the $\mathcal{T}$ size of 120k is large while that of 150k is small. As shown in Figure 8 (a), the corresponding time cost for generating a large $\mathcal{T}$ is high.

7. CONCLUSIONS

In this paper, we study the discovery of matching dependencies. First, we formally define the evaluation of matching dependencies by using support and confidence. Then, we introduce the problem of discovering the MDs with minimum confidence and support requirements. Both pruning strategies and approximation of the exact algorithm 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. Moreover, since MDs might not be able to express many matching rules by similarity thresholds, we propose the extended matching dependencies with similarity intervals. The eMDs discovery problem is to find a concise tableau of interval patterns with the minimum size. Due to the NP-completeness of the problem, we study the greedy algorithms. Advanced pruning techniques are also proposed to improve the efficiency, together with a proved bound of the pruning rate on average. The experimental evaluation demonstrates the performance of proposed methods.

Since this is the first work on discovering the matching dependencies, there are many aspects of work to develop in the future. For example, although the current approach can exclude the attributes that are not necessary to a MD, another issue is to minimize the number of attributes in the MD. However, the problem of determining attributes for FDs is already hard [19], where the matching similarity thresholds are not necessary to be considered. Moreover, the minimization problem of attributes can be introduced in eMDs as well, which is even more complicated with a pattern tableau instead of a single pattern of thresholds. Finally, and most importantly, more exiting applications of MDs and eMDs are expected to be explored in the future work.

8. REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[3] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[4] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *ICDE*, pages 746–755, 2007.

[5] L. Bravo, W. Fan, F. Geerts, and S. Ma. Increasing the expressivity of conditional functional dependencies without extra complexity. In *ICDE*, pages 516–525, 2008.

[6] L. Bravo, W. Fan, and S. Ma. Extending dependencies with conditions. In *VLDB*, pages 243–254, 2007.

[7] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.

[8] A. E. F. Clementi, P. Crescenzi, and G. Rossi. On the complexity of approximating colored-graph problems. In *COCOON*, pages 281–290, 1999.

[9] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD Conference*, pages 201–212, 1998.

[10] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *VLDB*, pages 315–326, 2007.

[11] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[12] W. Fan. Dependencies revisited for improving data quality. In *PODS*, pages 159–170, 2008.

[13] W. Fan, F. Geerts, L. V. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *ICDE*, 2009.

[14] W. Fan, S. Ma, Y. Hu, J. Liu, and Y. Wu. Propagating functional dependencies with conditions. *PVLDB*, 1(1):391–407, 2008.

[15] Full Version. Technique Report, CSE, HKUST. *http://www.cse.ust.hk/~sshaoxu/paper/dependency.html*.

[16] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[17] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *PVLDB*, 1(1):376–390, 2008.

[18] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an rdbms for web data integration. In *WWW*, pages 90–101, 2003.

[19] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[20] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, pages 169–178, 2000.

[21] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

APPENDIX

A. PROOF

A.1 Proof for Discovering MDs

PROOF OF LEMMA 1.

Let $\mathsf{cover}(\lambda_1)$ and $\mathsf{cover}(\lambda_2)$ denote the set of statistical tuples that satisfy the threshold λ_1 and λ_2 respectively, e.g., $\mathsf{cover}(\lambda_2) = \{s \mid s[X] \vDash \lambda_2, s \in \mathcal{D}\}$. 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 $s \in \mathsf{cover}(\lambda_2)$, 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., $\mathsf{cover}(\lambda_2) \subseteq \mathsf{cover}(\lambda_1)$. Referring to the definition of $\mathsf{support}$, we have $\mathsf{support}(\varphi_1) \geq \mathsf{support}(\varphi_2)$. □

PROOF OF LEMMA 2.

Since the first k tuples have $s_i[Y] \vDash \lambda_Y$, according to the computation of $P(X,Y)$ and $P(X)$, we have $\frac{P_i^j(X,Y)}{P_i^j(X)} = 1.0, 1 \leq i \leq k$. Moreover, for the remaining $n-k$ tuples with $s_i[Y] \nvDash \lambda_Y$, the $P(X,Y)$ value will not change any more, i.e., $P_i^j(X,Y) = P_k^j(X,Y), k+1 \leq i \leq n$. Meanwhile, the corresponding $P(X)$ is non-decreasing, that is, $P_k^j(X) \leq P_{i_1}^j(X) \leq P_{i_2}^j(X)$ for any $k+1 \leq i_1 < i_2 \leq n$. Consequently, we have $\frac{P_{i_1}^j(X,Y)}{P_{i_1}^j(X)} \geq \frac{P_{i_2}^j(X,Y)}{P_{i_2}^j(X)}$. □

PROOF OF LEMMA 3.

Let

$$\begin{aligned} \alpha &= P_k^j(X,Y) \\ \bar{\alpha} &= P_n^j(X,Y) - P_k^j(X,Y) \end{aligned}$$

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

$$Z = \frac{\alpha(\beta+\bar{\beta})}{\beta(\alpha+\bar{\alpha})} \leq 1 + \frac{\bar{\beta}}{\beta}$$

First, we have $\beta = \alpha + \sum_{i=1}^{k} s_i[P(X \vDash \lambda_j, Y \nvDash \lambda_Y)] \geq \alpha$. Note that α is the approximate support of the MD φ with matching similarity threshold pattern λ_j on the attributes X. According to the minimum support constraint, for a valid λ_j, we have $\beta \geq \alpha \geq \eta_s$. Thereby,

$$Z \leq 1 + \frac{\bar{\beta}}{\eta_s}$$

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

$$Z \leq 1 + \epsilon$$

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

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

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

$$Z \geq \frac{\beta+\bar{\beta}}{\beta + \frac{\bar{\beta}}{\eta_c}} = 1 - \frac{\bar{\beta}(1-\eta_c)}{\beta\eta_c + \bar{\beta}}$$

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

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

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

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

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 $S^n = \alpha + \bar{\alpha}$. Therefore,

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

Recall that we have $\alpha \geq \eta_s$ and $\bar{\alpha} \leq \bar{\beta} \leq \epsilon\eta_s$.

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

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

A.2 Proof of Discovering eMDs

PROOF OF THEOREM 2.

To prove that the optimal eMD generation problem belongs to NP, we can show that a pattern tableau $\mathcal{T}$ of any size can be verified in polynomial time, such that the support and confidence η_s, η_c are satisfied. Let n be the number of statistical tuples in $\mathcal{D}$, and let w be the number patterns in $\mathcal{T}$. According to the formula (3.2) and (3.2), we can compute the confidence and support of ψ in $\mathcal{O}(nw)$ time.

To prove that the optimal eMD generation problem is NP-hard, we can show that the 3-PARTITE GRAPH VERTEX COVER problem [8], which is NP-complete, is polynomial-time reducible to the optimal eMD generation problem. A *3-partite* graph is also known as *3-colorable* graph, where the vertices are partitioned into 3 groups and each edge has two vertices from two different partitions. Let G be a 3-partite graph with 3 partitions (A, B, C) and m edges among the partitions. Let a_i, b_j, c_k denote the vertices from partitions A, B, C respectively. The VERTEX COVER problem is to find a minimum set of vertices that covers all the m edges in the graph G.

We construct a distribution $\mathcal{D}$ with attributes A, B, C, D, having $X = [A, B, C]$ and $Y = [D]$ for eMD. Let $\mathsf{dom}(D) = \{0, 1\}$ and $\delta_Y = [1, 2)$. The values of each vertex is assigned from the domain of the corresponding partition (attribute), e.g., $a_i \in \mathsf{dom}(A)$. We assume that for any two vertices a_{i-1}, a_i, we have $a_{i-1} + 1 < a_i$. In other words, there is no continues values of intervals. Let a, b, c be the values having $a < a_i, b < b_j, c < c_k$ for all i, j, k. The statistical tuples in $\mathcal{D}$ are built as follows. For each edge (a_i, b_j), we construct two data tuples $\{a_i, b_j, c, 1, 1\}$ and $\{a_i + 1, b_j + 1, c, 0, 1)\}$ in $\mathcal{D}$, where $c \in \mathsf{dom}(C)$. Similarly, we also add tuples $\{a, b_j, c_k, 1, 1\}$, $\{a, b_j + 1, c_k + 1, 0, 1\}$ and $\{a_i, b, c_k, 1, 1\}$, $\{a_i + 1, b, c_k + 1, 0, 1\}$ for the edges (b_j, c_k) and (a_i, c_k) respectively, with $a \in \mathsf{dom}(A)$ and $b \in \mathsf{dom}(B)$. Finally, we have total $2m$ statistical tuples in $\mathcal{D}$ for the m edges in G.

Let $\eta_s = m$ and $\eta_c = 1$. A feasible solution always exists, that is, the tableau $\mathcal{T}'$ with patterns $\{[a_i, a_i+1), [b, d), [c, d)\}$, $\{[a, d), [b_j, b_j+1), [c, d)\}$ and $\{[a, d), [b, d), [c_k, c_k+1)\}$ for all i, j, k, where d is the domain size of attributes and each pattern tableau corresponds to a vertex. The patterns in $\mathcal{T}'$ covers exact the m data tuples in $\mathcal{D}$ with $D = 1$. By the relaxation of intervals in patterns, e.g., from $[a_i, a_i+1)$ to $[a_i, a_i+2)$ or higher, the tuple $(a_i+1, b_j+1, c, 0, 1)$ will be included and the confidence turns to be less than 1, which is not valid. Moreover, the relaxation of minimum threshold in a pattern, e.g., from $[b, d)$ to $[b-1, d)$, will still cover the same m statistical tuples, that is, equivalent to the original pattern. On the other hand, a tighten matching similarity threshold such as $[b+1, d)$ will exclude some data tuples like $(a_i, b, c_k, 1, 1)$ and reduce the support to be lower than m, which is also invalid. The relaxation or tightening the patterns turns to be either invalid or equivalent, therefore, we can always assume that the feasible solutions consist of patterns in $\mathcal{T}'$. The vertex cover problem transforms to find a minimum subset of patterns (vertices) $\mathcal{T} \subseteq \mathcal{T}'$ that covers at least m statistical tuples (edges) in $\mathcal{D}$. The reduction can be conducted in polynomial time.

To conclude, the optimal eMD generation problem with minimum tableau size is NP-complete. □

PROOF OF THEOREM 3.

The *k-partial set cover* problem is: given a set of n elements $E = \{E_1, E_2, \ldots, E_n\}$, a collection S of subsets of E, $S = \{S_1, S_2, \ldots, S_m\}$, a cost function of S, and a k, to find a minimum cost sub-collection of S, say $\mathcal{T}$, that covers at least k elements of E. Here, each E_i denotes a statistical tuple s and each S_j denotes a pattern δ in our problem. The k corresponds to the minimum support η_s, and the cost function counts the number of subsets, i.e., $|\mathcal{T}|$. According to the *partial covering* problem [16], the greedy algorithm has a $\ln n + 1$ approximation.

The $\arg\max$ operation can be implemented in a constant time by amortizing the support values into a constant domain. Note that the **for** statement in line 2 of Algorithm 5 adds a specific s to a certain δ exactly once, i.e., $\mathcal{O}(nc^2)$, and the **while** statement in line 3 of Algorithm 6 removes a specific s from a certain δ at most once. Thereby, the GA complexity is also $\mathcal{O}(nc^2)$. □

A.3 Proof of Pruning for eMDs

PROOF OF LEMMA 4.

Let s be any tuple in $\mathsf{cover}(\delta_2)$ with $s[X] \vDash \delta_2[X]$. For any attribute A, let $\delta_2[A] = [g, h)$ and $\delta_1[A] = [v, u)$. According to $s[A] \vDash \delta_2[A]$ and $\delta_1 \lessdot \delta_2$, we have $v \le g \le s[A] < g+h \le v+u$, that is, $s[A] \vDash \delta_1[A]$ for each attribute A as well. In other words, all the statistical tuples s in $\mathsf{cover}(\delta_2)$ are also contained in the δ_1's, i.e., $\mathsf{cover}(\delta_2) \subseteq \mathsf{cover}(\delta_1)$. Recalling the support definition, we have $\mathsf{support}(\delta_1) \ge \mathsf{support}(\delta_2)$.

Moreover, since $\mathsf{cover}(\delta_2) \subseteq \mathsf{cover}(\delta_1)$, we have $\mathsf{support}(\delta_1) = \mathsf{support}(\delta_2)$ if and only if $\mathsf{cover}(\delta_2) = \mathsf{cover}(\delta_1)$. That is, the patterns δ_1 and δ_2 cover exact the same statistical tuples. According to the greedy algorithm, there is no difference between the pattern δ_1 and δ_2, i.e., equivalent. □

PROOF OF THEOREM 4.

Let

$$\begin{aligned}\mathsf{tup}(\delta) &= \{s \mid s[X] \vDash \delta\} \\ &= \delta[A_1] \times \cdots \times \delta[A_m] \times \mathsf{dom}(\mathcal{D} \setminus X)\end{aligned}$$

define all the possible statistical tuples (may not appear in current $\mathcal{D}$) that can be covered by δ. Since, the set of $\mathsf{cover}(\delta)$ for pattern δ records all the statistical tuples $s \in \mathcal{D}$ that $s[X] \vDash \delta$, we have $\mathsf{cover}(\delta) \subseteq \mathsf{tup}(\delta)$.

We first prove the pruning of I_1. Let δ be the pattern with the interval $[\alpha, \alpha+u)$ on attribute A, i.e., $\delta[A] = [\alpha, \alpha+u)$. Moreover, for $u = 2, 3, \ldots$, let δ_1 and δ_2 have the same interval with δ on all the attributes except $\delta_1[A] = [\alpha, \alpha+1)$ and $\delta_2[A] = [\alpha+1, \alpha+u)$, having $\delta \lessdot \delta_1$ and $\delta \lessdot \delta_2$. According to the definition of $\delta[A]$, we have $\delta[A] = \delta_1[A] \cup \delta_2[A]$ and $\delta_1[A] \cap \delta_2[A] = \emptyset$. Since δ_1 and δ_2 share the same intervals with δ on all the other attributes, we also have $\mathsf{tup}(\delta) = \mathsf{tup}(\delta_1) \cup \mathsf{tup}(\delta_2)$ and $\mathsf{tup}(\delta_1) \cap \mathsf{tup}(\delta_2) = \emptyset$. Therefore, for the $\mathsf{cover}(\delta) \subseteq \mathsf{tup}(\delta)$, we can split it into $\mathsf{cover}(\delta) = \mathsf{cover}(\delta_1) \cup \mathsf{cover}(\delta_2)$ as well. Since α does not appear in $\mathcal{D}$, we have $\mathsf{cover}(\delta_1) = \emptyset$, i.e., $u = 1$ of I_1. In addition, we also have $\mathsf{cover}(\delta) = \emptyset \cup \mathsf{cover}(\delta_2) = \mathsf{cover}(\delta_2)$ and $\delta \lessdot \delta_2$. According to Lemma 4, the pattern δ with interval $[\alpha, \alpha+u)$ on A is equivalent to δ_2 with $[\alpha+1, \alpha+u)$ having $\mathsf{support}(\delta) = \mathsf{support}(\delta_2)$. In other words, the patterns δ with interval $[\alpha, \alpha+u)$ are duplicates and can be pruned. Thus, $I_1 = \{[\alpha, \alpha+u) \mid u = 1, 2, \ldots\}$ can be pruned.

We then prove the pruning of I_2. Similarly, for a pattern δ with the interval $[\alpha-u+1, \alpha+1)$ on attribute A, let $\delta_1[A] = [\alpha-u+1, \alpha)$ and $\delta_2[A] = [\alpha, \alpha+1)$. We can also have $\mathsf{cover}(\delta) = \mathsf{cover}(\delta_1) \cup \mathsf{cover}(\delta_2)$. Recall that $\mathsf{cover}(\delta_2) = \emptyset$ due to the interval $[\alpha, \alpha+1)$ on attribute A of δ_2, i.e., $u = 1$ of I_2. Moreover, according to Lemma 4, the pattern δ with interval $[\alpha-u+1, \alpha+1)$ is equivalent to δ_1 with $[\alpha-u+1, \alpha)$, for $u = 2, 3, \ldots$, and can be pruned as duplicates. Consequently, $I_2 = \{[\alpha-u+1, \alpha+1) \mid u = 1, 2, \ldots\}$ can be pruned. □

PROOF OF LEMMA 5.

We first prove no updates on C_5. Let A be the attribute having $[0, v) \lessdot \delta'[A]$ in $B_5[A]$. Since $[0, v) = \{0, 1, \ldots, v-1\}$ and $[v, u) = \{v, v+1, \ldots, v+u-1\}$, we have $[0, v) \cap [v, u) = \emptyset$, that is, $\delta'[A] \cap \delta[A] = \emptyset$ as well. Referring to the product operator in the $\mathsf{tup}(\delta)$ definition, we can infer $\mathsf{tup}(\delta') \cap \mathsf{tup}(\delta) = \emptyset$. Also, for cover sets of the patterns $\mathsf{cover}(\delta) \subseteq \mathsf{tup}(\delta)$, the intersection is $\mathsf{cover}(\delta') \cap \mathsf{cover}(\delta) = \emptyset$. In other words, the operation $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$ takes no effect on pattern δ'. The patterns in C_5 are not updated.

Similarly, we have $[v+u, d) = \{v+u, v+u+1, \ldots, d-1\}$ for $B_6[A]$. According to $[v+u, d) \cap [v, u) = \emptyset$, we can also infer $\mathsf{cover}(\delta') \cap \mathsf{cover}(\delta) = \emptyset$ for $\delta' \in C_6$. Thus, the patterns in C_6 are not updated. □

PROOF OF THEOREM 5.

For a pattern $\delta' \in C_p$, the interval $\delta'[A]$ of any attribute A can either come from $B_2[A]$, $B_3[A]$ or $B_4[A]$. Let

$$\begin{aligned}C_2 &= \{\delta' \mid \exists A, \delta'[A] \in B_2[A], \delta' \in C_p\} \\ C_3 &= \{\delta' \mid \exists A, \delta'[A] \in B_3[A], \delta' \in C_p\} \\ C_4 &= \{\delta' \mid \forall A, \delta'[A] \in B_4[A], \delta' \in C_p\}\end{aligned}$$

having $C_p = C_2 \cup C_3 \cup C_4$.

We first prove the pruning of $\delta' \in C_2$ with $\delta'[A] = [g, h) \in B_2[A]$. Let δ_1 and δ_2 have the same intervals with δ' on all the attributes except $\delta_1[A] = [g, v)$ and $\delta_2[A] = [v, g+h)$, having $\delta' \lessdot \delta_1$, $\delta' \lessdot \delta_2$. According to the definition of $\delta'[A]$, we have $\delta'[A] = \delta_1[A] \cup \delta_2[A]$ and $\delta_1[A] \cap \delta_2[A] = \emptyset$. Since δ_1 and δ_2 share the same intervals with δ' on all the other attributes, we also have $\mathsf{tup}(\delta') = \mathsf{tup}(\delta_1) \cup \mathsf{tup}(\delta_2)$ and $\mathsf{tup}(\delta_1) \cap \mathsf{tup}(\delta_2) = \emptyset$. Therefore, for the $\mathsf{cover}(\delta') \subseteq \mathsf{tup}(\delta')$, we can split it into $\mathsf{cover}(\delta') = \mathsf{cover}(\delta_1) \cup \mathsf{cover}(\delta_2)$ as well. Since $\delta_2 \in C_p$, we can prune δ_2 with $\mathsf{cover}(\delta_2) = \emptyset$. Thereby, according to Lemma 4, the pattern δ' is equivalent to δ_1 with $\mathsf{cover}(\delta') = \mathsf{cover}(\delta_1)$ after the current

elimination step. Moreover, we have $[0, v) \lessdot \delta_1[A] = [g, v)$, that is, $\delta_1[A] \in B_5[A]$ and $\delta_1 \in C_5$. According to Lemma 5, δ_1 already exists in $\mathcal{C}_e$ without updates in the current iteration. Therefore, the pattern δ' can be pruned as duplicates after the current elimination step in $\mathcal{C}_e$.

Next, we prove the pruning of $\delta' \in C_3$ with $\delta'[A] = [g, h) \in B_3[A]$. Let δ_1 and δ_2 have the same intervals with δ' on all the attributes except $\delta_1[A] = [g, v + u)$ and $\delta_2[A] = [v + u, g + h)$. Similarly, we can also split the cover set of δ' into $\mathsf{cover}(\delta') = \mathsf{cover}(\delta_1) \cup \mathsf{cover}(\delta_2)$. Since $\delta_1 \in C_p$ can be pruned with $\mathsf{cover}(\delta_1) = \emptyset$, δ' is equivalent to δ_2. In addition, $\delta_2 \in C_6$ already exists in current $\mathcal{C}_e$ without update. Therefore, the pattern $\delta' \in C_3$ can be pruned as duplicates as well.

Finally, we prove the pruning of $\delta' \in C_4$. For any attribute A, according to the definition of $B_4[A]$, we have $\delta[A] \lessdot \delta'[A]$, that is, $\delta \lessdot \delta'$ for each $\delta' \in C_4$. Referring to Lemma 4, we have $\mathsf{cover}(\delta') \subseteq \mathsf{cover}(\delta)$. After the elimination step $\mathsf{cover}(\delta') = \mathsf{cover}(\delta') \setminus \mathsf{cover}(\delta)$, $\mathsf{cover}(\delta') = \emptyset$ and δ' can be pruned. □

B. TABLE

Table 6: Reference of cosine similarity of names in Contacts

No.	name1	name2	sim(name1, name2)
1	Claire Green	Claire Gree_	0.95
2	Claire Greem	Claire Gree_	0.95
3	Claire Green	Claire Greem	0.90
4	J. Smith	W. J. Smith	0.85
5	Jason Smith	J. Smith	0.60
6	Jason Smith	W. J. Smith	0.50
7	Claire Green	W. J. Smith	0
⋮	⋮	⋮	⋮
15	Claire Gree_	Jason Smith	0