2012 IEEE 28th International Conference on Data Engineering

Parameter-Free Determination of Distance Thresholds for Metric Distance Constraints

Shaoxu Song [#1], Lei Chen [*2], Hong Cheng [$3]

[#] Key Laboratory for Information System Security, Ministry of Education; TNList; School of Software, Tsinghua University
[1] sxsong@tsinghua.edu.cn

[*] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology
[2] leichen@cse.ust.hk

[$] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong
[3] hcheng@se.cuhk.edu.hk

***Abstract*—The importance of introducing distance constraints to data dependencies, such as differential dependencies (DDs) [28], has recently been recognized. The metric distance constraints are tolerant to small variations, which enable them apply to wide data quality checking applications, such as detecting data violations. However, the determination of distance thresholds for the metric distance constraints is non-trivial. It often relies on a truth data instance which embeds the distance constraints. To find useful distance threshold patterns from data, there are several guidelines of statistical measures to specify, e.g., support, confidence and dependent quality. Unfortunately, given a data instance, users might not have any knowledge about the data distribution, thus it is very challenging to set the right parameters. In this paper, we study the determination of distance thresholds for metric distance constraints, in a parameter-free style. Specifically, we compute an expected utility based on the statistical measures from the data. According to our analysis as well as experimental verification, distance threshold patterns with higher expected utility could offer better usage in real applications, such as violation detection. We then develop efficient algorithms to determine the distance thresholds having the maximum expected utility. Finally, our extensive experimental evaluation demonstrates the effectiveness and efficiency of the proposed methods.**

I. Introduction

The data collected from different sources are often dirty, including inconsistencies, conflicts and violations, due to various errors introduced by humans and machines (see [3] for a survey). Recently, *functional dependencies* (FDs) have been revisited and revised with extensions [5] to capture the inconsistency in the dirty data [10]. For example, the following functional dependency fd_1 over the Hotel relation specifies a constraint that for any two tuples in Hotel, if they have the same Address, then their Region values must be equal.

$$\mathsf{fd}_1 \quad : \quad [\mathsf{Address}] \rightarrow [\mathsf{Region}]$$

The constraints are useful in detecting data violations, a very important task for data cleaning [9]. For instance, we can use the above fd_1 to detect violations in an instance of Hotel in Table I. For the tuples t_5 and t_6 with the equal value on Address, they have different values of Region, which are then treated as a violation of the above fd_1.

Unfortunately, real-world information often has various representation formats. The strict equality function limits the usage of FDs (as well as the extensions that are still based on the equality). For example, according to fd_1, the tuples t_1 and t_2 in Table I will be detected as a "violation", since they have "different" Region values but agree on Address. However, "Chicago" and "Chicago, IL" denote the same region in the real-world with different representation formats, which are not violations. Moreover, t_4 and t_6, which have similar Address but different Regions, are true violations. Unfortunately, they cannot be detected by fd_1, since their address values are not exactly equal.

TABLE I
Example of Hotel

ID	Name	Address	Region	
01	West Wood Hotel	Fifth Avenue, 61st Street	Chicago	t_1
01	West Wood	Fifth Avenue, 61st Street	Chicago, IL	t_2
01	West Wood (61)	5th Avenue, 61st St.	Chicago, IL	t_3
22	St. Regis Hotel	No.3, West Lake Road.	Boston, MA	t_4
22	St. Regis Hotel	#3, West Lake Rd.	Boston	t_5
22	St. Regis	#3, West Lake Rd.	Chicago, MA	t_6

To address small variations in data formats, functional dependencies have recently been extended by incorporating distance constraints, namely *differential dependencies* (DDs) [28]. Informally, DDs declare the dependencies between the determinant attributes X and the dependent attributes Y, $X \rightarrow Y$, with respect to metric distances, such as edit distance (see [4] for a survey of distance/similarity metrics). In contrast to the equality function on each attribute as FDs, a DD can specify a pattern φ of distance thresholds on attributes of X and Y. For example, in Hotel, we may have a DD as

$$\mathsf{dd}_1 \quad : \quad ([\mathsf{Address}] \rightarrow [\mathsf{Region}], < 8, 3 >)$$

where $< 8, 3 >$ is a pattern φ of distance thresholds on Address and Region respectively. It states a constraint on metric distance: for any two tuples from Hotel, if they have distance on Address less than a threshold (≤ 8), then their Region values should be similar as well, i.e., the edit distance on Region is less than the corresponding threshold (≤ 3).

This study focuses on DDs as a general type of metric distance constraints, which employ distance metrics on both

1084-4627/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDE.2012.46

sides of attributes. There are some other notations that specify distance metrics only in one side [19], [12] and could be regarded as special cases. Indeed, when all the thresholds are set to 0, i.e., equality, DDs have the same semantics as FDs. For instance, the above fd_1 can be represented by a DD ($[\mathsf{Address}] \rightarrow [\mathsf{Region}], < 0, 0 >$). Thereby, FDs are also special cases of DDs.

Motivation: Unlike FDs, which already imply the equality function, it is rather difficult to manually determine the proper settings of distance thresholds for metric distance constraints. For instance, a DD with a very tight threshold (e.g., close to 0 as FDs) will be too strict to be tolerant to various information formats, while a loose threshold (e.g., close to $d_{\max}$ the maximum distance value) is meaningless since any data can satisfy it. In light of the dependency discovery from data [17], we can also rely on a truth data instance to determine the distance thresholds. The truth data instance is clean without violations and embeds the distance constraints.

The problem studied in this paper is then to determine the distance thresholds φ for metric distance constraints from the truth data. Given the attributes X, Y on a data instance, there are numerous different distance thresholds to choose for the attributes in DDs. Clearly, not all the settings of thresholds are useful. Following the evaluation of FDs [18], we may also study the measures for DDs, in order to indicate how reliable and useful a metric distance constraint is. As opposed to the equality function in the previous work, the major difference of measures for metric distance constraints is about the tolerance via distance metrics.

Specifically, the utility of metric distance constraints could be investigated by certain statistical measures including *support*, *confidence* [6] and the unique *dependent quality*. Let $\varphi[X]$ and $\varphi[Y]$ be the projections of thresholds on attributes X and Y of a distance threshold pattern φ.

i) The *support* of φ is the proportion of tuple pairs in the truth data whose distances satisfy the thresholds in $\varphi[XY]$ on both attributes of X and Y. When applying the metric distance constraints to detect violations in the dirty data, a φ with *high support* is preferred in order to detect more violations.

ii) The *confidence* of φ is the ratio of tuple pairs satisfying $\varphi[XY]$ to the pairs satisfying $\varphi[X]$. Note that the confidence measure is analogous to the precision of violation detection. Thereby, a φ with *high confidence* is preferred.

iii) The *dependent quality* of φ denotes the quality of tolerance on the dependent attributes Y. It indicates how close the distance threshold $\varphi[Y]$ to the equality is. As shown in the following example, if the dependent quality is low (i.e., $\varphi[Y]$ is far away from equality), the constraint is meaningless and useless.[1]

First, if the dependent quality is set too high, e.g., $\varphi[Y] = 0$, which is exactly the equality function in FDs, then the constraint could be too strict and may identify violations by mistake as illustrated in the previous example. Consequently, the confidence measure will be low. On the other hand, consider a φ with the lowest dependent quality, i.e., with threshold $\varphi[Y] = d_{\max}$ the maximum distance value. It has the highest confidence 1.0, since any tuple pairs can always have distances $\leq d_{\max}$ on Y. Unfortunately, such a φ would miss all the violations and is useless. For example, we consider ($[\mathsf{Address}] \rightarrow [\mathsf{Region}], < 8, d_{\max} >$), whose threshold on the dependent attribute Region is $d_{\max}$. Since any pair of tuples always has distance on Region $\leq d_{\max}$, the confidence of this DD is the highest 1.0. However, t_4 and t_6 in Table I cannot be detected by such a DD, while these two tuples are true violations and can be detected by dd_1.

Recognize that real applications such as violation detection need metric distance constraints with high statistical measures, i.e., high support, high confidence and high dependent quality at the same time. A straight-forward idea is to specify the minimum requirements of these three statistical measures by users, in the determination of distance thresholds. Unfortunately, given a data instance, users may have no idea about the data distribution. Without any prior knowledge, it might be difficult to set the parameters of minimum support, confidence and dependent quality, respectively. As illustrated in the above examples, setting the requirements of some measures too high will make the others low.

In this work, we propose methods to determine distance thresholds in a parameter-free style. Intuitively, our approach targets on automatically returning those "best" φ, i.e., not existing any other settings that can be found having higher support, confidence, and dependent quality than the returned results at the same time. Most important of all, we verified that these automatically found "best" distance threshold patterns are indeed more effective than other randomly selected settings (including FDs) in the application of violation detection.

We further notice that automatically determining the "best" settings of distance thresholds is non-trivial in terms of computation cost. Indeed, the determination process has to consider the combination of distance thresholds in the attributes. Therefore, we explore several pruning opportunities to speed up the determination process.

Contributions: To sum up, we make the following contributions in this work.

i) We propose the expected utility of distance threshold patterns, such that higher support, confidence and dependent quality will yield a higher expected utility.

ii) We develop efficient pruning algorithms for the distance threshold determination, together with several advanced pruning bounds with respect to the expected utility.

iii) We conduct an extensive experimental evaluation over three real data sets. In particular, we evaluate the effectiveness of returned results in the violation detection application. The experiments also demonstrate that our pruning strategies can significantly improve the efficiency of determination.

The remainder of this paper is organized as follows. First, we introduce some related work in Section II and the preliminary of this study in Section III. Section IV develops the computation of expected utility, together with analysis on its properties. In Section V, we present pruning algorithms. Our

[1] A large $\varphi[X]$ could be meaningful. See Section III-A for a discussion.

extensive experimental evaluation is reported in Section VI. Finally, Section VII concludes this paper.

II. Related Work

The notation of differential dependencies (DDs) is first proposed in [28], together with the corresponding fundamental issues such as reasoning. The discovery in [28] targets on a minimal cover of all DDs that exactly hold in a data instance, while the statistical measures for DDs with respect to the utility are not studied. Such statistical measures are essential to tell the importance of a DD. Therefore, in this paper, we first introduce three statistical measures, i.e., support, confidence and dependent quality, for DDs. Then, we study the distance threshold determination problem regarding the statistical measures, in particular in a parameter-free style.

Besides the differential dependencies considered in this paper, other notations of metric distance constraints are also studied. Koudas et al. [19] propose the *metric functional dependencies* (MFDs), which employ distance metrics in the dependent attributes Y but have the equality function in the determinant side X. Therefore, MFDs can be regarded as special cases of DDs, and the threshold determination techniques proposed in this study can be directly applied to MFDs. Fan et al. [12] introduce the *matching dependencies* (MDs) for record matching, another important aspect of data cleaning. Instead of detecting data violations as DDs and MFDs do, MDs aim to identify the duplicates based on the similarity on certain attributes X. In addition, comparable dependencies (CDs) over heterogeneous data are also studied in [29], which need the more complicated schema mapping support. We believe that the proposed threshold determination techniques could be useful and possibly extended for determining MDs or CDs in the future work. Indeed, the discovery of data dependencies from a data instance has been widely studied [21], [26], [20]. Unfortunately, since the equality function is usually considered, these previous works can hardly address the determination of distance thresholds for metric distance constraints. The most related work is [27] about MDs discovery. It differs from this work in two aspects: (1) the determination on metric distance only needs to be considered in the determinant side X for MDs; (2) there is no issue about tolerance to address for dependent attributes in MDs. Most importantly, the previous work needs to specify the parameters for the measures manually, while the problem introduced in this study is to determine the distance thresholds in a parameter-free style.

A dependency rule can be measured in various ways. In the measures of FDs [17], g_3 measure [18] is widely used, that is, the minimum number of tuples that have to be removed from the relation instance for the FD to hold. The computation of g_3 measure relies on grouping tuples by equal values, which cannot be applied to distance metrics on tuple pairs. The confidence and support measures are also used in evaluating FDs [6] and conditional FDs [14], [7], [11]. The confidence can be interpreted as an estimate of the probability that a randomly drawn pair of tuples agreeing on X also agrees on Y [6]. In our study, we also utilize support and confidence for metric distance constraints, which are defined based on tuple pairs. The major difference of measures for metric distance constraints to the previous work is about dependent quality. As introduced in the introduction, metric distance constraints need an additional measure to evaluate the quality of tolerance in the dependent attributes.

TABLE II
Notations

Symbol	Description
φ	Pattern of distance thresholds
$d_{\max}$	Maximum distance value
$\mathcal{R}$	Original data relation
$\mathcal{M}$	Matching relation with matching tuples
$\mathcal{C}$	Candidate set of distance threshold patterns DDs
$\mathrm{D}(\varphi)$	LHS support of $\varphi[X]$
$\mathrm{S}(\varphi)$	Support of φ
$\mathrm{C}(\varphi)$	Confidence of φ
$\mathrm{Q}(\varphi)$	Dependent quality of φ
$\bar{\mathrm{U}}(\varphi)$	Expected utility of φ

Instead of setting minimum requirements of several measures, in association rule mining [1], it is also studied to return the most interesting rules according to the specified measures. For example, Han et al. [16] indicate that setting minimum support requirement is quite subtle: a too small threshold may lead to the generation of thousands of patterns, whereas a too big one may often generate no answers. Webb and Zhang [31] study the k-optimal rule discovery, where the leverage is used as a single value measure instead of support and confidence separately. Scheffer [25] studies the trade-off between support and confidence for finding association rules, by computing an expected prediction accuracy. Compared with the previous work on mining association rules, our problem for metric distance constraints is different and more challenging in two aspects: (1) the support and confidence measures are defined with respect to the distance in tuple pairs, instead of the group (set) of identical items in association rules; (2) besides support and confidence, we have to balance the additional measure of dependent quality on distance, which is a concept that does not exist in association rules. Consequently, our expected utility needs to consider more measures, i.e., support, confidence and dependent quality, which has more complicated computation and properties as illustrated in Section IV.

III. Problem Statement

In this section, we first introduce the formal definition and statistical measures for metric distance constraints, which raises the problem of determining thresholds with high utility. Table II lists the frequently used notations in this paper.

A. Preliminary

Consider a relation $\mathcal{R}$. Let $X \subseteq \mathcal{R}$ denote a set of *determinant attributes*, and $Y \subseteq \mathcal{R}$ be a set of *dependent attributes*. For each attribute $A \in X \cup Y$, we associate a

distance metric d_A, e.g., edit distance [24] or cosine similarity [8] on text, or the metric on numeric data [2]. The selection of distance metric is out of the scope of this study. Please refer to [4] for a survey. Let $d_A(a_1, a_2)$ denote the distance between two values a_1 and a_2 on attribute A.

A *differential dependency* (DD) [28], $(X \rightarrow Y, \varphi)$, specifies a *pattern φ of distance thresholds* on attributes X and Y, also denoted by $\varphi[XY]$. The projection $\varphi[A]$ on attribute $A \in X \cup Y$ denotes the distance threshold of A.

A DD states the constraint that, for any two tuples t_1 and t_2 in a relation instance r of $\mathcal{R}$, if $d_{A_i}(t_1[A_i], t_2[A_i]) \leq \varphi[A_i], \forall A_i \in X$, then it must have $d_{A_j}(t_1[A_j], t_2[A_j]) \leq \varphi[A_j], \forall A_j \in Y$, where $\varphi[A_i]$ and $\varphi[A_j]$ are distance thresholds on the attributes of A_i and A_j respectively.

For example, a DD $([\mathsf{Address}] \rightarrow [\mathsf{Region}], < 8, 3 >)$ in the Hotel relation specifies the constraint that if two tuples have similar Address (with edit distance no greater than $\varphi[\mathsf{Address}] = 8$) then their Region values are also similar (with edit distance $\leq \varphi[\mathsf{Region}] = 3$).

Note that the distance thresholds on the determinant attributes X could be large. If $\varphi[X]$ is close to $d_{\max}$ the maximum distance value, it states that no matter whether the X values are similar or not, the constraints on Y can be achieved anyway. The larger the $\varphi[X]$ is, the weaker the Y side depends on the similarity of X. When $\varphi[X] = d_{\max}$, the constraint on X is unlimited, i.e., any tuple pair can always have distance $\leq d_{\max}$. In fact, for any attribute $A \in \mathcal{R} \setminus (X \cup Y)$ that does not appear in a DD $(X \rightarrow Y, \varphi)$, it already implies an unlimited constraint $\varphi[A] = d_{\max}$ on A. We say that $\varphi[Y]$ is *independent* of A.

B. Statistical Measures

In order to compute the measures of support, confidence and dependent quality, we define the following statistics. Following the support and confidence measures defined on tuple pairs [6], we study the statistics of tuple pairs with respect to metric distance. Given an instance of relation $\mathcal{R}$ with N *data tuples*, we conduct a pair-wise matching of all N tuples. The metric distance of each tuple pair is denoted by a *matching tuple* b, where $b[A]$ is the distance of the data tuple pair on attribute $A \in \mathcal{R}$. As illustrated below, for the evaluation of each φ, we have to visit all the tuple pairs once. To avoid the recomputation among the evaluation of different φ, we can precompute the matching tuples from all tuple pairs and store them for reuse.

Let $\mathcal{M}$ be a matching relation with total $M = |\mathcal{M}| = \frac{N(N-1)}{2}$ matching tuples obtained from the pair-wise matching. If the distance values on X of a matching tuple b can satisfy the corresponding thresholds $\varphi[X]$, we say that b satisfies $\varphi[X]$, denoted by $b \vDash \varphi[X]$. Let $\mathsf{count}(b \vDash \varphi[X])$ denote the total number of matching tuples $b \in \mathcal{M}$ that can satisfy $\varphi[X]$. We introduce the following statistics.

$$\mathrm{D}(\varphi) = \frac{\mathsf{count}(b \vDash \varphi[X])}{M} \tag{1}$$

Indeed, $\mathrm{D}(\varphi)$ can be regarded as the support of $\varphi[X]$ in the left-hand-side of φ. To illustrate the computation of statistical measures, we consider the previously used running example in Table I in the introduction. Since there are 15 tuple pairs in Table I and 6 of them have distance ≤ 8 on Address, we have $\mathrm{D}(\mathsf{dd}_1) = 6/15 = 0.4$.

$$\mathrm{C}(\varphi) = \frac{\mathsf{count}(b \vDash \varphi[XY])}{\mathsf{count}(b \vDash \varphi[X])} \tag{2}$$

Moreover, $\mathrm{C}(\varphi)$ is exactly the *confidence* definition of φ, i.e., the ratio of tuple pairs satisfying $\varphi[XY]$ to the pairs satisfying $\varphi[X]$. For example, in Table I, among 6 pairs of tuples satisfying $\varphi[X]$ of dd_1, there are 4 tuple pairs also having Region distances ≤ 3. Thereby, the confidence of dd_1 is $4/6$. Consequently, the overall *support* of φ is $\mathrm{S}(\varphi) = \mathrm{C}(\varphi)\mathrm{D}(\varphi)$, i.e., the proportion of tuple pairs whose distances satisfy the thresholds in $\varphi[XY]$ on both attributes of X and Y, e.g., $\mathrm{S}(\mathsf{dd}_1) = 4/15$.

$$\mathrm{Q}(\varphi) = \frac{\sum_{A \in Y} d_{\max} - \varphi[A]}{|Y| d_{\max}} \tag{3}$$

Finally, the *dependent quality* $\mathrm{Q}(\varphi)$ denotes the quality of tolerance in the dependent side, i.e., how the distance thresholds $\varphi[Y]$ on dependent attributes Y is close to the equality. Intuitively, as presented in formula 3, the smaller the distance thresholds on Y are, the higher the dependent quality $\mathrm{Q}(\varphi)$ is. When the distance thresholds are set to the smallest 0, the dependent quality will be the highest 1.0, i.e., the equality case. On the other hand, the largest threshold $d_{\max}$ will have the lowest dependent quality 0.0.

C. Threshold Determination Problem

There are various distance threshold settings that can be associated to the DDs on $X \rightarrow Y$, while only some of them with high utility are interesting to users, i.e., those ones with high support, confidence and dependent quality. The determination process often needs users to specify the minimum requirements of these three statistical measures.

As mentioned in the introduction, it is difficult to set parameters of minimum support, confidence and dependent quality respectively. Setting some requirements high may lead other measures to be low. To avoid tuning parameters manually, we are interested in an overall evaluation of utility.

Let b be any matching tuple. We study the following prediction probability

$$\mathrm{U}(\varphi) = \Pr(b \vDash \varphi[Y], \mathrm{Q}(\varphi) \text{ is high} \mid b \vDash \varphi[X]), \tag{4}$$

i.e., the conditional probability of b satisfying $\varphi[Y]$ with high dependent quality given b satisfies $\varphi[X]$. Intuitively, to accurately detect the violations with small distance, we expect the above probability of a φ to be high. This $\mathrm{U}(\varphi)$ can roughly denote the utility of confidence and dependent quality, while support is not investigated.

Therefore, we aim to compute an *expected utility* to refine $\mathrm{U}(\varphi)$ w.r.t. confidence and dependent quality by using support,

$$\bar{\mathrm{U}}(\varphi) = E(\mathrm{U}(\varphi) \mid \mathrm{C}(\varphi), \mathrm{D}(\varphi), \mathrm{Q}(\varphi)),$$

where $\mathrm{C}(\varphi), \mathrm{D}(\varphi)$ and $\mathrm{Q}(\varphi)$ are the statistics observed from the matching relation $\mathcal{M}$. The computation and properties of the expected utility are discussed in the following Section IV. In short, we can find that the expected utility of a φ is high if it has high support, confidence and dependent quality.

Finally, we formulate the determination problem without manually specifying the individual requirements.

Definition 1. *The distance threshold determination problem is to find a distance threshold pattern φ for the* DD *on $X \rightarrow Y$ with the maximum expected utility $\bar{\mathrm{U}}(\varphi)$.*

IV. Computing Expected Utility

In this section, we present the detailed computation steps of the expected utility, and then analyze the corresponding properties.

Computation: Given a data instance, we can compute the statistics $\mathrm{C}(\varphi), \mathrm{D}(\varphi), \mathrm{Q}(\varphi)$ denoted by $\mathrm{C}, \mathrm{D}, \mathrm{Q}$. According to the Bayesian rule, we have the expected utility as follows,

$$\begin{aligned}
\bar{\mathrm{U}}(\varphi) &= E(\mathrm{U} \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) \\
&= \int u P(\mathrm{U} = u \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) du \\
&= \int u \frac{P(\mathrm{U} = u, \mathrm{C}, \mathrm{D}, \mathrm{Q})}{P(\mathrm{C}, \mathrm{D}, \mathrm{Q})} du \\
&= \int u \frac{P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D})}{P(\mathrm{C}, \mathrm{Q} \mid \mathrm{D})} du \\
&= \frac{\int u P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D}) du}{P(\mathrm{C}, \mathrm{Q} \mid \mathrm{D})}.
\end{aligned}$$

Consider all possible u values,

$$\begin{aligned}
& \int P(\mathrm{U} = u \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) du = 1 \\
\Leftrightarrow \quad & \int \frac{P(\mathrm{U} = u, \mathrm{C}, \mathrm{D}, \mathrm{Q})}{P(\mathrm{C}, \mathrm{D}, \mathrm{Q})} du = 1 \\
\Leftrightarrow \quad & \int \frac{P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D})}{P(\mathrm{C}, \mathrm{Q} \mid \mathrm{D})} du = 1 \\
\Leftrightarrow \quad & P(\mathrm{C}, \mathrm{Q} \mid \mathrm{D}) = \int P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D}) du.
\end{aligned}$$

Based on the above two derivations, we have

$$E(\mathrm{U} \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) = \frac{\int u P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D}) du}{\int P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D}) P(\mathrm{U} = u \mid \mathrm{D}) du}.$$

$P(\mathrm{C}, \mathrm{Q} \mid \mathrm{U} = u, \mathrm{D})$ can be modeled exactly as a Binomial distribution [23], denoted by $\mathrm{DCQ} \sim B(\mathrm{D}, u)$. Recall the definition of Binomial distribution: Consider a set of n objects, each of which yields success with probability p, then the probability of k successes in n objects is given by the *probability mass function* of Binomial distribution:

$$f(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}.$$

In our scenario, we find D instances from the matching relation $\mathcal{M}$ satisfying $\varphi[X]$ (i.e., $n = \mathrm{D}$). Among them, we observed DCQ instances satisfying $\varphi[Y]$ with high dependent quality (i.e., $k = \mathrm{DCQ}$ successes). Moreover, according to formula 4, u is the probability of predicting Y with high dependent quality given X (i.e., $p = u$). Finally, the probability of observing C, Q is given by $f(\mathrm{DCQ}; \mathrm{D}, u)$. Thereby, we can rewrite the computation formula,

$$E(\mathrm{U} \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) = \frac{\int u f(\mathrm{DCQ}; \mathrm{D}, u) P(\mathrm{U} = u \mid \mathrm{D}) du}{\int f(\mathrm{DCQ}; \mathrm{D}, u) P(\mathrm{U} = u \mid \mathrm{D}) du}.$$

$P(\mathrm{U} = u \mid \mathrm{D})$ is exactly $P(\mathrm{U} = u)$, since the utility U is independent of D which concerns X only. The prior $P(\mathrm{U} = u)$ denotes the distribution of φ, i.e., the fraction of φ with $\mathrm{U} = u$. Note that the observed CQ can be interpreted as an estimation of the prediction probability of utility u, i.e., the probability of matching tuples that satisfy $\varphi[X]$ also satisfying $\varphi[Y]$ with a high dependent quality. Thus, we use the histogram $P(\mathrm{CQ})$ to estimate $P(\mathrm{U} = u)$, which can also be modeled by the Binomial distribution,

$$P(\mathrm{U} = u) = \pi(u) = f(u; 1, \overline{\mathrm{CQ}}),$$

where $\overline{\mathrm{CQ}}$ is the mean of CQ.

Finally, we can compute the expected utility $\bar{\mathrm{U}}(\varphi)$ by

$$\bar{\mathrm{U}}(\varphi) = E(\mathrm{U} \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) = \frac{\int u f(\mathrm{DCQ}; \mathrm{D}, u) \pi(u) du}{\int f(\mathrm{DCQ}; \mathrm{D}, u) \pi(u) du}. \quad (5)$$

Property: We discuss the properties of the expected utility, that is, how the support, confidence and dependent quality measures of φ contribute to the expected utility $\bar{\mathrm{U}}(\varphi)$. (All the proofs of the following lemma, theorem and proposition can be found in the full version [13].)

Theorem 1. *For any φ_1, φ_2, if φ_1 has higher support than φ_2, denoted by $\frac{\mathrm{S}(\varphi_1)}{\mathrm{S}(\varphi_2)} = \rho, \rho \geq 1$, and the confidence and dependent quality of φ_1 are higher than those of φ_2 as follows $\frac{\mathrm{C}(\varphi_1)}{\mathrm{C}(\varphi_2)} \geq \rho, \frac{\mathrm{Q}(\varphi_1)}{\mathrm{Q}(\varphi_2)} \geq \frac{1}{\rho}$, then we have $\bar{\mathrm{U}}(\varphi_1) \geq \bar{\mathrm{U}}(\varphi_2)$.*

This conclusion verifies our intuition that higher support, confidence and dependent quality will contribute to a larger expected utility. Indeed, to clarify the contributions of measures clearly, we can fix one of them, and derive the contributions of the other two measures. For instance, as we will see in Theorem 2 below, if any φ_1, φ_2 have the same D measure, then the higher confidence and dependent quality of φ_1 will yield a higher expected utility $\bar{\mathrm{U}}(\varphi_1)$ than that of φ_2.

So far, we have presented the computation of expected utility with clarification on the corresponding properties. We are now ready to find the distance threshold patterns for DDs with the maximum expected utility in a parameter-free style, instead of studying how to specify the minimum requirements of several statistical measures.

V. Determination Algorithm

In this section, we study the finding of one (or several) setting of distance thresholds for the DDs on $X \rightarrow Y$ with the maximum expected utility, i.e., $\varphi_{\max} = \arg\max_{\varphi} \bar{\mathrm{U}}(\varphi)$. The determination process has mainly two steps: **(i)** to find the best $\varphi[Y]$ when given a fixed $\varphi[X]$; **(ii)** to find the desired $\varphi[X]$ together with its best $\varphi[Y]$ which has the maximum $\bar{\mathrm{U}}(\varphi)$.

The previous FDs discovery considers the combination of attributes [17] and targets on minimizing a single measure, such as g_3 [18]. In contrast, we study the determination of distance thresholds on certain attributes X, Y and aim to maximize the expected utility with respect to three measures. Recognizing the major difference and challenges, in the following, we investigate several pruning methods and bounds, by exploring the unique features of the expected utility.

A. Determination for Dependent Attributes

First, we consider the determination for the dependent attributes: given a fixed $\varphi[X]$, to find the corresponding best $\varphi[Y]$ on the dependent attributes Y with the maximum $\bar{\mathrm{U}}(\varphi)$.

For an attribute $A \in Y$, we can consider the search space of distance threshold $\varphi[A]$ from 0 to $d_{\max}$. For example, supposing that the maximum distance value is 9, we denote all the distance thresholds as $\mathsf{dis} = \{0, 1, 2, \ldots, 9\}$.

Let $\mathcal{C}_Y$ denote the set of distance threshold patterns $\varphi[Y]$ by enumerating all the distance thresholds $\varphi[A] \in \mathsf{dis}$ for all the dependent attributes $A \in Y$. For instance, as shown in Figure 1 (a), each node, such as $< 1, 1 >$, corresponds to a $\varphi[Y] \in \mathcal{C}_Y$. Suppose that two attributes in Y have the same $\mathsf{dis} = \{0, 1, 2, \ldots, 9\}$. Figure 1 (a) illustrates all the distance threshold patterns in $\mathcal{C}_Y$, from $< 0, 0 >$ to $< 9, 9 >$.

Since $\varphi[X]$ on the determinant attributes X is fixed in the current step, according to formula 1, the $\mathrm{D}(\varphi)$ value is the same for any φ such that $\varphi[Y] \in \mathcal{C}_Y$. Therefore, we mainly study the other two measures $\mathrm{C}(\varphi)$ and $\mathrm{Q}(\varphi)$ in terms of contributions to the expected utility $\bar{\mathrm{U}}(\varphi)$.

Theorem 2. *Consider any two φ_1, φ_2, having the same $\mathrm{D}(\varphi_1) = \mathrm{D}(\varphi_2) = \mathrm{D}$. If their confidence and dependent quality satisfy $\mathrm{C}(\varphi_1)\mathrm{Q}(\varphi_1) \geq \mathrm{C}(\varphi_2)\mathrm{Q}(\varphi_2)$, then we have $\bar{\mathrm{U}}(\varphi_1) \geq \bar{\mathrm{U}}(\varphi_2)$.*

According to Theorem 2, for a fixed $\varphi[X]$, to find a φ with the maximum $\bar{\mathrm{U}}(\varphi)$ is equivalent to find the one with the maximum $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$. For each $\varphi[Y] \in \mathcal{C}_Y$, we can directly calculate the dependent quality $\mathrm{Q}(\varphi)$ by formula 3. Moreover, together with the given $\varphi[X]$ on the determinant attributes X, we can also compute the measure $\mathrm{C}(\varphi)$ from $\mathcal{M}$ using formula 2. The result φ with the maximum $\bar{\mathrm{U}}(\varphi)$ can be found by one pass through all the $\varphi[Y] \in \mathcal{C}_Y$.

Algorithm 1 presents the approach of finding the best $\varphi[Y]$ for a given $\varphi[X]$ such that the expected utility $\bar{\mathrm{U}}(\varphi)$ is maximized. As mentioned, to find the result, we only need to compute $\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$ for each φ_i such that $\varphi_i[Y] \in \mathcal{C}_Y$ and $\varphi_i[X] = \varphi[X]$. Line 3 in Algorithm 1 computes the $\mathrm{C}(\varphi_i)$ from $\mathcal{M}$ by using formula 2. Let $V_{\max}$ denote the maximum value of $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$ in the first $i-1$ candidates, i.e.,

$$V_{\max} = \max_{j=1}^{i-1} \mathrm{C}(\varphi_j)\mathrm{Q}(\varphi_j).$$

Initially, we can set $V_{\max} = 0$. The $\varphi_{\max}$ in Line 5 records the distance threshold pattern with the maximum $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$ value, and will be returned as the result.

Suppose that we consider d values of distance thresholds in each attribute, i.e., $|\mathsf{dis}| = d$. In this PA approach, we need to consider the combination of d distance thresholds in all Y attributes as the candidate patterns in $\mathcal{C}_Y$, i.e., $|\mathcal{C}_Y| = d^{|Y|}$ in size. For each candidate φ, there is a costly step to compute $\mathrm{C}(\varphi)$. According to formula 2, for each φ, we have to count the number of matching tuples in $\mathcal{M}$ which can satisfy $\varphi[XY]$. With the increase of data sizes, the matching relation size $M = |\mathcal{M}|$ will be large as well. The cost $\mathcal{O}(Md^{|Y|})$ of PA is high. Therefore, we propose the following pruning techniques to reduce the number of candidates during the computation.

Algorithm 1 for Dependent Attributes
PA$(\mathcal{M}, \varphi[X], V_{\max})$

1: **for** each $\varphi_i[Y] \in \mathcal{C}_Y$ **do**
2: let $\varphi_i[X] = \varphi[X]$
3: compute $\mathrm{C}(\varphi_i)$ from $\mathcal{M}$
4: **if** $\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i) > V_{\max}$ **then**
5: $\varphi_{\max} = \varphi_i, V_{\max} = \mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$
6: **return** $\varphi_{\max}$

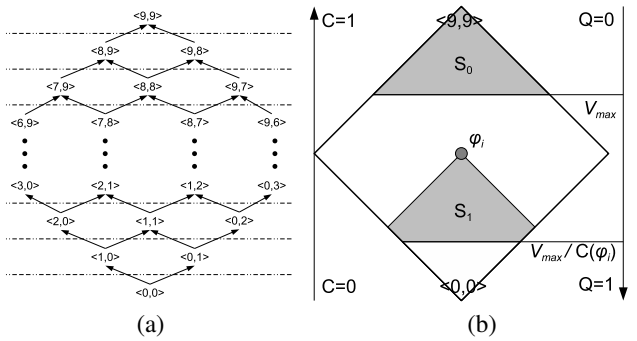

Fig. 1. Pruning on dependent attributes

Pruning approach: To study the pruning of pattern candidates, let's first introduce the relationships between distance threshold patterns. Consider any attribute set Z of $\mathcal{R}$.

Definition 2. *For any φ_1, φ_2, if $\varphi_1[A] \geq \varphi_2[A]$ holds for all the attributes, $\forall A \in Z$, then we say that $\varphi_1[Z]$* dominates *$\varphi_2[Z]$, denoted by $\varphi_1[Z] \preccurlyeq \varphi_2[Z]$.*

Figure 1 (a) shows the dominant relationships of candidates in $\mathcal{C}_Y$. An arrow from a node $\varphi_2[Y]$ to $\varphi_1[Y]$ denotes that $\varphi_1[Y]$ dominates $\varphi_2[Y]$, i.e., $\varphi_1[Y] \preccurlyeq \varphi_2[Y]$. We present the dominant relationships between two neighbor levels. For the domination between other levels, we can derive them by the transitivity property, i.e., if $\varphi_1[Y] \preccurlyeq \varphi_2[Y]$ and $\varphi_2[Y] \preccurlyeq \varphi_3[Y]$, we have $\varphi_1[Y] \preccurlyeq \varphi_3[Y]$ as well. For example, in Figure 1 (a), $< 1, 1 >$ also dominates $< 0, 0 >$, which is not shown.

Lemma 1. *For any two φ_1, φ_2, having $\varphi_1[X] = \varphi_2[X]$ and $\varphi_1[Y] \preccurlyeq \varphi_2[Y]$, then $\mathrm{C}(\varphi_1) \geq \mathrm{C}(\varphi_2)$ and $\mathrm{Q}(\varphi_1) \leq \mathrm{Q}(\varphi_2)$.*

According to Lemma 1, by a downward traversal of candidates in the dominant graph, the dependent quality increases from 0 to 1. On the other hand, as shown in Figure 1 (b), the confidence increases from 0 to 1 in an upward traversal.

Consider the current φ_i in traversal of $\mathcal{C}_Y$. Let $\varphi_{\max}$ denote

the candidate with the maximum $\mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max}) = V_{\max}$ in the previously processed $i-1$ candidates in $\mathcal{C}_Y$,

$$\varphi_{\max} = \arg\max_{\varphi_j, j\in[1,i-1]} \mathrm{C}(\varphi_j)\mathrm{Q}(\varphi_j).$$

We have the opportunity of pruning the following two sets of remaining φ_k without conducting the costly computation of $\mathrm{C}(\varphi_k)$:

i) *Pruning by* $\varphi_{\max}$. The first pruning opportunity is introduced by $\varphi_{\max}$ of the previously processed $i-1$ candidates.

Proposition 1. *For any* $\varphi_k[Y] \in \mathcal{C}_Y$ *with* $\mathrm{Q}(\varphi_k) \le V_{\max}$*, we have* $\bar{\mathrm{U}}(\varphi_{\max}) \ge \bar{\mathrm{U}}(\varphi_k)$.

We identify the candidates that can be pruned according to Proposition 1. Let

$$S_0 = \{\varphi_k \mid \mathrm{Q}(\varphi_k) \le V_{\max}, \varphi_k[Y] \in \mathcal{C}_Y\}.$$

Since confidence is always less than or equal to 1, we have $\mathrm{C}(\varphi_k)\mathrm{Q}(\varphi_k) \le \mathrm{Q}(\varphi_k) \le V_{\max}$. As illustrated in Figure 1 (b), all the φ_k in S_0 can be safely pruned without computing $\mathrm{C}(\varphi_k)$ from $\mathcal{M}$.

ii) *Pruning by* φ_i. The second pruning opportunity is developed according to the current φ_i in i-th step.

Proposition 2. *For any* $\varphi_k[Y] \in \mathcal{C}_Y$ *with* $\varphi_i[Y] \lessdot \varphi_k[Y]$ *and* $\mathrm{Q}(\varphi_k) \le \frac{V_{\max}}{\mathrm{C}(\varphi_i)}$*, we have* $\bar{\mathrm{U}}(\varphi_{\max}) \ge \bar{\mathrm{U}}(\varphi_k)$.

Similarly, the candidates that can be pruned according to Proposition 2 are represented by

$$S_1 = \{\varphi_k \mid \varphi_i \lessdot \varphi_k, \mathrm{Q}(\varphi_k) \le \frac{V_{\max}}{\mathrm{C}(\varphi_i)}, \varphi_k[Y] \in \mathcal{C}_Y\},$$

as shown in Figure 1 (b). According to $\varphi_i[Y] \lessdot \varphi_k[Y]$, we have $\mathrm{C}(\varphi_k) \le \mathrm{C}(\varphi_i)$. It implies $\mathrm{C}(\varphi_k)\mathrm{Q}(\varphi_k) \le \mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_k) \le V_{\max}$. Therefore, all the φ_k in S_1 can be safely pruned as well, without computing $\mathrm{C}(\varphi_k)$.

Based on the above two propositions, we develop Algorithm 2, namely PAP, which prunes the candidates of S_0 and S_1 when passing through each $\varphi_i \in \mathcal{C}_Y$. The operation prune$(\varphi, q)$ removes all the patterns dominated by φ from the candidate set $\mathcal{C}_Y$, whose dependent quality $\mathrm{Q}(\varphi_k)$ is less than or equal to q. For example, in Line 7, prune$(\varphi_i, \frac{V_{\max}}{\mathrm{C}(\varphi_i)})$ removes patterns φ in S_1 from $\mathcal{C}_Y$, according to Proposition 2. Note that any φ_k is dominated by φ_0, $\varphi_0[Y] \lessdot \varphi_k[Y]$, where φ_0 has $\varphi_0[A] = d_{\max}$ for each attribute $A \in Y$. Therefore, in Line 6, we can also use the same function prune$(\varphi_0, V_{\max})$ to prune the set S_0, according to Proposition 1.

Finally, we discuss the processing orders of candidates in $\mathcal{C}_Y$. Heuristically, in order to maximize the pruning power, we would like to find the largest $V_{\max}$ as early as possible. As illustrated in Figure 1, if we first process patterns in the top, the corresponding dependent quality is small (≈ 0), which leads to a small $V_{\max}$, indicating weak pruning power. On the other hand, when the patterns in the bottom are processed first, the corresponding $V_{\max}$ is also small due to the low confidence (≈ 0). Therefore, during the processing, we heuristically choose the patterns in the middle to be processed first, namely *mid-first* order in $\mathcal{C}_Y$.

Algorithm 2 for Dependent Attributes with Pruning
PAP$(\mathcal{M}, \varphi[X], V_{\max})$

1: **for** each $\varphi_i[Y] \in \mathcal{C}_Y$ **do**
2: let $\varphi_i[X] = \varphi[X]$
3: compute $\mathrm{C}(\varphi_i)$ from $\mathcal{M}$
4: **if** $\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i) > V_{\max}$ **then**
5: $\varphi_{\max} = \varphi_i, V_{\max} = \mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$
6: prune$(\varphi_0, V_{\max})$ {remove patterns in S_0 from $\mathcal{C}_Y$}
7: prune$(\varphi_i, \frac{V_{\max}}{\mathrm{C}(\varphi_i)})$ {remove patterns in S_1 from $\mathcal{C}_Y$}
8: **return** $\varphi_{\max}$

B. Determination for Determinant Attributes

Next, we consider all possible distance threshold patterns of the determinant attributes X, say $\mathcal{C}_X$, and find a φ with the maximum $\bar{\mathrm{U}}(\varphi)$. The straight-forward approach is to compute the best $\varphi[Y]$ for each $\varphi[X] \in \mathcal{C}_X$ respectively by using PA, and then return the one with the maximum $\bar{\mathrm{U}}(\varphi)$.

Algorithm 3 presents the straight-forward computation of a distance threshold $\varphi_{\max}$ with the maximum expected utility. For each $\varphi_i[X] \in \mathcal{C}_X$, Line 3 computes $\mathrm{D}(\varphi_i)$ from $\mathcal{M}$ by using formula 1. By calling the PA or PAP algorithm, we can compute the best $\varphi_i[Y]$ with respect to the current $\varphi_i[X]$. For the initial bound $V_{\max}$, as mentioned before, we can set $V_{\max} = 0$ as illustrated in Line 4. Finally, the expected utility $\bar{\mathrm{U}}(\varphi_i)$ of each φ_i in Line 5 is computed by using formula 5 as presented in Section IV. Here, $U_{\max}$ records the maximum expected utility in the first $i-1$ candidates,

$$U_{\max} = \max_{j=1}^{i-1} \bar{\mathrm{U}}(\varphi_j).$$

Initially, we set $U_{\max} = 0$. The returned $\varphi_{\max}$ is the distance threshold pattern with the maximum expected utility $U_{\max}$.

Algorithm 3 for Determinant Attributes
DA$(\mathcal{M})$

1: $U_{\max} = 0$
2: **for** each $\varphi_i[X] \in \mathcal{C}_X$ **do**
3: compute $\mathrm{D}(\varphi_i)$ from $\mathcal{M}$
4: compute $\varphi_i[Y]$ by PA$(\mathcal{M}, \varphi_i[X], 0)$
5: $\bar{\mathrm{U}}(\varphi_i) = E(\mathrm{U}(\varphi_i) \mid \mathrm{C}(\varphi_i), \mathrm{D}(\varphi_i), \mathrm{Q}(\varphi_i))$
6: **if** $U_{\max} < \bar{\mathrm{U}}(\varphi_i)$ **then**
7: $\varphi_{\max} = \varphi_i, U_{\max} = \bar{\mathrm{U}}(\varphi_i)$
8: **return** $\varphi_{\max}$

Again, considering all d distance threshold values in each attribute, there are $d^{|X|}$ and $d^{|Y|}$ candidates in set $\mathcal{C}_X$ and $\mathcal{C}_Y$ respectively. Let $c = |X| + |Y|$ be the total number of attributes in X and Y. The total number of candidates evaluated in the algorithm is $|\mathcal{C}_X \times \mathcal{C}_Y| = d^c$. Note that for each candidate, we have to traverse the matching relation $\mathcal{M}$ in order to compute corresponding measures such as confidence. Therefore, the entire complexity of the approaches, e.g., DA+PA, is $\mathcal{O}(Md^c)$, where M is the total number of matching tuples in $\mathcal{M}$. Note that pruning version PAP can be applied to replace the basic approach PA, namely DA+PAP.

Advanced pruning bound using $\varphi_{\max}$: The most costly part of the above DA algorithm is still the computation of $\varphi_i[Y]$ by using PA or PAP, which traverses $\mathcal{M}$ frequently. As mentioned, in order to improve the pruning power of PAP, we expect to find a larger pruning bound $V_{\max}$. Let's first study the following properties in terms of the expected utility.

Theorem 3. *Consider any two φ_1, φ_2, having $\mathrm{D}(\varphi_1) \geq \mathrm{D}(\varphi_2)$. If their confidence and dependent quality satisfy*

$$\mathrm{C}(\varphi_2)\mathrm{Q}(\varphi_2) \leq 1 - \frac{\mathrm{D}(\varphi_1)}{\mathrm{D}(\varphi_2)}\Big(1 - \mathrm{C}(\varphi_1)\mathrm{Q}(\varphi_1)\Big)$$

then we have $\bar{\mathrm{U}}(\varphi_1) \geq \bar{\mathrm{U}}(\varphi_2)$.

Intuitively, we can prune those φ_2 whose $\mathrm{C}(\varphi_2)\mathrm{Q}(\varphi_2)$ is no higher than $1 - \frac{\mathrm{D}(\varphi_1)}{\mathrm{D}(\varphi_2)}\Big(1 - \mathrm{C}(\varphi_1)\mathrm{Q}(\varphi_1)\Big)$, i.e., the new pruning bound $V_{\max}$. To apply this pruning bound, we require a precondition $\mathrm{D}(\varphi_1) \geq \mathrm{D}(\varphi_2)$.

Therefore, we can adopt an ordering of candidates in $\mathcal{C}_X$ having $\mathrm{D}(\varphi_{i_1}) \geq \mathrm{D}(\varphi_{i_2})$ for any $i_1 < i_2$. The ordering can be done in linear time by amortizing the $\mathrm{D}(\varphi)$ values into a constant domain. In the following algorithms, we assume that the candidates in $\mathcal{C}_X$ have already been listed in descending order of $\mathrm{D}(\varphi)$ values.

Let $\varphi_{\max}$ be the current result with the maximum expected utility by evaluating the first $i-1$ candidates in $\mathcal{C}_X$. Since we process $\mathcal{C}_X$ in descending order of $\mathrm{D}(\varphi)$ values, for the next φ_i, we have $\mathrm{D}(\varphi_{\max}) \geq \mathrm{D}(\varphi_i)$. According to Theorem 3, we can compute an advanced pruning bound

$$V_{\max} = 1 - \frac{\mathrm{D}(\varphi_{\max})}{\mathrm{D}(\varphi_i)}\Big(1 - \mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max})\Big) \qquad (6)$$

for the computation of $\varphi_i[Y]$ by using PAP, in the current φ_i.

Algorithm 4 extends Algorithm 3 by introducing the advanced pruning bound $V_{\max}$ instead of simply assigning $V_{\max} = 0$ in the original DA. We compute the advanced pruning bound in Line 4 by using formula 6, and use this $V_{\max}$ for pruning in the determination on dependent attributes Y (Line 5). Note that all candidates in $\mathcal{C}_Y$ might be pruned in PAP by the advanced $V_{\max}$, i.e., no $\varphi_i[Y]$ returned (Line 6). Then, the current $\varphi_i[X]$ can be discarded safely without computing the corresponding $\bar{\mathrm{U}}(\varphi_i)$.

Algorithm 4 for Determinant Attributes with Pruning DAP($\mathcal{M}$)

```
1:  U_max = 0
2:  for each φ_i[X] ∈ C_X do
3:     compute D(φ_i) from M
4:     V_max = 1 − (D(φ_max)/D(φ_i)) (1 − C(φ_max)Q(φ_max))
5:     compute φ_i[Y] by PAP(M, φ_i[X], V_max)
6:     if φ_i[Y] exists then
7:        Ū(φ_i) = E(U(φ_i) | C(φ_i), D(φ_i), Q(φ_i))
8:        if U_max < Ū(φ_i) then
9:           φ_max = φ_i, U_max = Ū(φ_i)
10: return φ_max
```

Recall that when the advanced pruning bound is not available, i.e., $V_{\max} = 0$ initially in DA+PAP, we use a mid-first order to process candidates in $\mathcal{C}_Y$ in order to find a large $V_{\max}$ as early as possible. However, we now already calculate an advanced pruning bound $V_{\max} > 0$ in DAP. Heuristically, in order to prune more candidates, we would like to process those ones in $\mathcal{C}_Y$ which can dominate more candidates, according to the prune operator in PAP. For example, we can use a *top-first* order, which first processes candidates in the top, since the top candidates (e.g., $< 9, 9 >$ in Figure 1 (a)) always dominate the bottom ones. Therefore, in the DAP+PAP approach with advanced pruning bounds, the top-first processing order of candidates in $\mathcal{C}_Y$ is preferred to achieve better time performance, which is also verified by our experiment results in Table V.

If the calculated bound $V_{\max}$ is less than 0, we can simply assign 0 to it. Once the bound is $V_{\max} > 0$, it can achieve a tighter pruning bound. Therefore, *practically*, the worst case of DAP is exactly the basic DA algorithm. In fact, as we can find, Theorem 2 is a special case of Theorem 3 when $\mathrm{D}(\varphi_1) = \mathrm{D}(\varphi_2)$. In other words, *theoretically*, the advanced pruning (e.g., DAP+PAP) developed based on Theorem 3 is a generalization of the basic pruning DA+PAP based on Theorem 2. Our experiments in Section VI also verify that DAP+PAP is at least no worse than DA+PAP.

Algorithm Extensions: When users require more than one answer for specific applications, the proposed algorithms can be easily extended to find the distance threshold patterns with second or third or l-th largest expected utilities. Specifically, instead of the maximum $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$, we use $V_{\max}$ to denote the l-th maximum $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$ in Algorithms 1, 2 and so on. Then, the return results are a set of l patterns of distance thresholds with the largest expected utilities. Since we relax $V_{\max}$ from the 1st maximum to l-th maximum, the pruning power will be weaker with the increase of l. We report the time performance on different answer sizes l in the experiments.

VI. Experimental Evaluation

We now report the experimental evaluation of the proposed methods. 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. We use three real data sets. The *Cora*[2] data set, prepared by McCallum et al. [22], consists of records of research papers, such as author, title, year, publisher, etc. The *Restaurant*[3] data set consists of restaurant records including attributes name, address, city and type. The *CiteSeer*[4] data set is selected with attributes including address, affiliation, subject, description, etc. During the preprocessing, we use edit distance with q-grams [15] to evaluate the distance of tuples in the original data. After pair-wised computation, we have up to 1,000,000 matching tuples in the matching relation $\mathcal{M}$ for each data set. The proposed techniques are then evaluated on the prepared matching relation $\mathcal{M}$.

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

TABLE III
EFFECTIVENESS OF EXAMPLE RESULTS FROM RULE 1

	$\varphi[X]$		$\varphi[Y]$		Measures				Violation Detection		
	author	title	venue	year	$S(\varphi)$	$C(\varphi)$	$Q(\varphi)$	$\bar{U}(\varphi)$	Precision	Recall	F-measure
φ_1	4	1	3	1	0.1529	0.3760	0.80	0.2325	0.3725	0.5425	0.4418
φ_2	5	2	3	1	0.1764	0.3667	0.80	0.2296	0.3718	0.6266	0.4667
φ_3	5	1	3	2	0.1632	0.3774	0.75	0.2232	0.3179	0.4492	0.3723
φ_4	4	2	3	2	0.1657	0.3657	0.75	0.2188	0.3073	0.4457	0.3638
φ_5	4	1	4	2	0.1529	0.3852	0.70	0.2108	0.2654	0.3267	0.2928
φ_6	5	2	5	2	0.1764	0.3985	0.65	0.2106	0.2459	0.3337	0.2831
fd	0	0	0	0	0.0064	0.3595	1.00	0.1064	0.4315	0.0735	0.1256

TABLE IV
EFFECTIVENESS OF EXAMPLE RESULTS FROM RULE 3

	$\varphi[X]$		$\varphi[Y]$		Measures				Violation Detection		
	name	address	city	type	$S(\varphi)$	$C(\varphi)$	$Q(\varphi)$	$\bar{U}(\varphi)$	Precision	Recall	F-measure
φ_1	10	7	5	10	0.2396	0.7967	0.25	0.1666	0.4889	0.3267	0.3917
φ_2	10	8	5	10	0.3467	0.7040	0.25	0.1577	0.3650	0.4137	0.3879
φ_3	10	7	1	10	0.2396	0.4106	0.45	0.1454	0.3305	0.3274	0.3289
φ_4	10	9	5	10	0.4709	0.6001	0.25	0.1344	0.2663	0.4782	0.3421
φ_5	10	8	2	10	0.3467	0.3640	0.40	0.1267	0.2675	0.4137	0.3249
φ_6	10	6	3	10	0.1333	0.4711	0.35	0.1182	0.4212	0.2082	0.2787
fd	0	0	0	0	0.0002	0.1219	1.00	0.0611	0.4062	0.0091	0.0178

In the experiments, suppose that users want to determine the distance thresholds for the metric distance constraints on the following $X \rightarrow Y$ attributes,

$Rule1 : cora(\mathsf{author}, \mathsf{title} \rightarrow \mathsf{venue}, \mathsf{year})$

$Rule2 : cora(\mathsf{venue} \rightarrow \mathsf{address}, \mathsf{publisher}, \mathsf{editor})$

$Rule3 : restaurant(\mathsf{name}, \mathsf{address} \rightarrow \mathsf{city}, \mathsf{type})$

$Rule4 : citeseer(\mathsf{address}, \mathsf{affiliation}, \mathsf{description} \rightarrow \mathsf{subject})$

where Rule 2 has a larger Y while Rule 4 has a larger X.

A. Result Study

The first experiment illustrates some example results of distance threshold patterns, and evaluates the effectiveness of applying them in the application of violation detection. To evaluate the detection accuracy, we use the measures of precision, recall and f-measure [30]. Let truth be the set of tuple pairs with violations that are manually inserted in random. Let found be the set of tuple pairs detected by applying the DDs. We have *precision* $= \frac{|\mathsf{truth} \cap \mathsf{found}|}{|\mathsf{found}|}$, *recall* $= \frac{|\mathsf{truth} \cap \mathsf{found}|}{|\mathsf{truth}|}$, and *f-measure*$= 2 \cdot \frac{precision \cdot recall}{precision + recall}$. The precision measure denotes the soundness and the recall means the completeness, while the f-measure is the overall accuracy. It is natural that higher precision, recall and f-measure are preferred.

Tables III and IV illustrate the example results on Rules 1 and 3, respectively. Each row denotes a φ with distance thresholds $\varphi[X], \varphi[Y]$ on attributes of X and Y respectively. We present the corresponding measures of support $S(\varphi)$, confidence $C(\varphi)$, dependent quality $Q(\varphi)$, as well as the expected utility $\bar{U}(\varphi)$. The results φ_i are listed in the descending order of $\bar{U}(\varphi)$. In the last row, we also report the corresponding FDs, where the distance threshold is 0 for each attribute.

Note that the results in Table IV show the interesting case of independence. The distance thresholds of attributes name and type are 10 in all the results, which is considered as the maximum distance value. It implies that there is no clear dependency relationship with respect to the name and type of Restaurants, while the similarity of city could be dependent on the similarity of address. By applying these DDs, we cannot detect the violations on type. Moreover, the similarity of name could not help in detecting the violations in city. Nevertheless, Table IV presents the best results in terms of the expected utility over the address and city attributes, since no dependency could be found on name and type.

The results also verify our property analysis of expected utility. According to Theorem 1, if a φ (e.g., with φ_2 in Table III) has higher support, confidence and dependent quality than another (e.g., φ_4 in Table III) at the same time, then the expected utility of φ_2 must be higher than that of φ_4. As observed in Table III, we have $\bar{U}(\varphi_2) \geq \bar{U}(\varphi_4)$, which verifies our analysis of expected utility properties in Section IV. Consequently, it is ensured that there does not exist any φ which has higher support, confidence and dependent quality at the same time than the returned φ_1 with the maximum expected utility.

The application of violation detection demonstrates the effectiveness of expected utility. As shown in Tables III and IV, the overall accuracy of detection (f-measure) approximately decreases with the decrease of the expected utility $\bar{U}(\varphi)$. It indicates that the expected utility can reflect the usefulness of

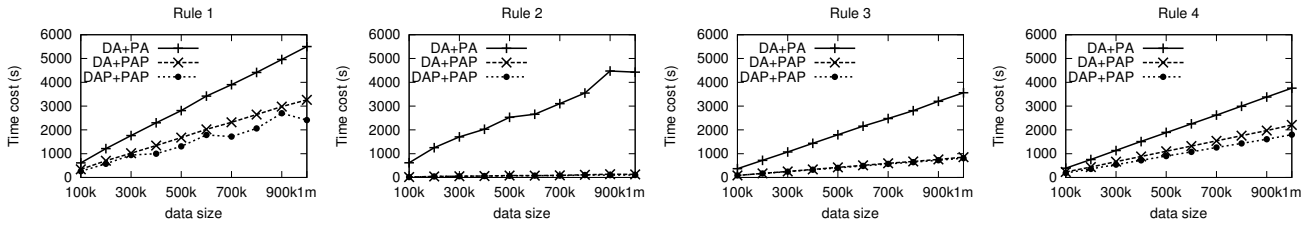

Fig. 2. Time performance on various data sizes (return largest $\bar{\mathrm{U}}$)

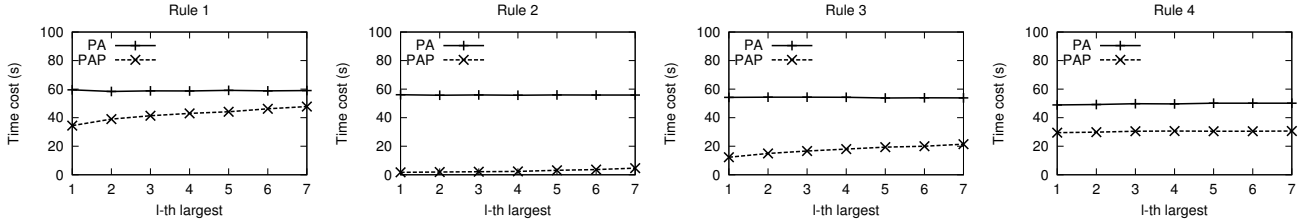

Fig. 3. Generation for dependent attributes

metric distance constraints in the application. Although some of the measures of a φ are higher, e.g., φ_6 in Table III with support 0.1764 and confidence 0.3985, the detection accuracy (f-measure) is still low due to the poor dependent quality of φ_6. This result also confirms that manually setting requirements of three measures could miss better results.

Finally, as shown in Tables III and IV, although the dependent quality is high in FD, due to the low support, the expected utility of FD is lower than DDs. Consequently, the detection effectiveness (f-measure) of FD is poor.

B. An Overview of Time Performance

Next, we evaluate the time performance of the proposed determination and pruning techniques. Since our proposed pruning techniques are proved (in fact also observed in the experiments) to be safe without missing answers or introducing errors, the returned results by different approaches are exactly the same. Therefore, in the following, we focus on the efficiency evaluation of the proposed techniques on various data sizes, up to 1,000,000 matching tuples. The compared approaches include the basic DA+PA, the basic pruning DA+PAP, the advanced pruning DAP+PAP. Note that advanced pruning bounds have no contribution to the algorithm PA without pruning techniques. Thus, approaches such as DAP+PA are equivalent to DA+PA, and omitted in this evaluation. Please refer to Section VI-C below for the additional results on evaluating the individual methods, such as PA vs. PAP.

Figure 2 reports the time cost of returning answers with the largest $\bar{\mathrm{U}}$. First, the time cost of approaches increases linearly with the data size, which shows the scalability of our determination methods. Although some specific tests may vary slightly in time cost due to the different data distribution and pruning power, the linear trend can still be clearly observed. This linear result with respect to the number of matching tuples M is not surprising, according to the complexity of determination algorithms $\mathcal{O}(Md^c)$ as illustrated in Section V.

Moreover, our pruning techniques work well for different rules and can reduce the time cost significantly in all data sizes. **i)** The pruning approach PAP of determination for dependent attributes shows lower time costs than the basic one PA. For example, in Rule 1, with the same DA method, the DA+PAP shows significantly lower time cost than the original DA+PA. **ii)** The DAP+PAP approach can provide a pruning bound that is at least no worse than the DA+PAP one. For instance, under the same PAP, the results in Rule 1 show better performance of DAP than DA, while Rule 3 verifies that the DAP is at least no worse than the DA. These results demonstrate the efficiency of the proposed pruning techniques, especially the superiority of the advanced DAP+PAP approach.

C. Evaluation on Specific Steps

In this experiment, we demonstrate the detailed results by applying different methods in specific steps. As mentioned in Section V, our algorithms can be easily extended to find the l-largest expected utility answers, which offer more options to users. Therefore, the following additional experiments also evaluate the performance on various l-th largest results, including the 1st largest.

Determination for Dependent Attributes: We first study the PA and PAP algorithms for determination in dependent attributes Y in Figure 3. The basic PA algorithm has to traverse all the candidates on Y without any pruning. Therefore, as shown in Figure 3, no matter how many l answers are requested, the PA approach has the same time cost. Meanwhile, the PAP approach can significantly reduce time cost by pruning candidates based on Propositions 1 and 2.

On the other hand, however, with the increase of the answer size l, the corresponding l-th largest expected utility value decreases, that is, the pruning bound is relaxed and the pruning power is weaker. Therefore, as presented in Figure 3, time cost of PAP increases with a larger answer size l.

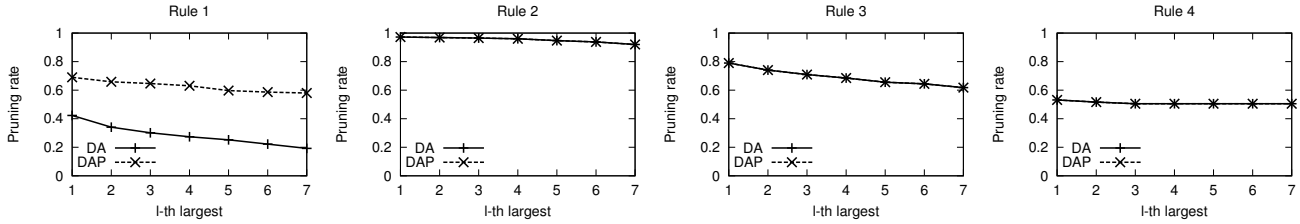

Fig. 4. Pruning power evaluation

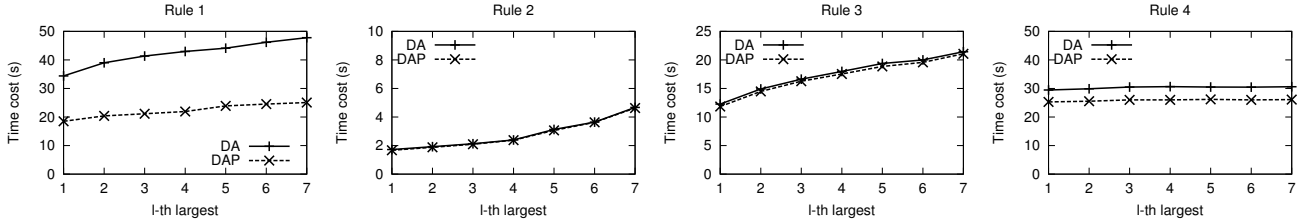

Fig. 5. Generation for determinant attributes

Moreover, the pruning power of the PAP algorithm is affected by the confidence and dependent quality measures, i.e., affected by data distribution. Therefore, PAP shows different improvement of time cost in four rules tests. In particular, Rule 2, which has more attributes in the dependent side, may have more opportunities of pruning by PAP. Consequently, as shown in Figure 3, the PAP can achieve a significant improvement in Rule 2. In fact, as discussed in the following experiments, the pruning bounds calculated by different techniques may show different performance as well.

Determination for Determinant Attributes: Recall that algorithms for determinant attributes, DA and DAP, contribute different pruning bounds. In order to evaluate the pruning power of these approaches, Figure 4 observes the pruning rate, i.e., the proportion of candidates that can avoid computation. For example, a pruning rate 0.9 denotes that 90% candidates can be safely pruned without computation. Obviously, the higher the pruning rate is, the lower time cost will be. We illustrate the time performance of DA and DAP algorithms for determinant attributes X in Figure 5.

First, the basic DA algorithm initially assigns $V_{\max} = 0$ as the pruning bound, while the advanced DAP uses the current $\varphi_{\max}$ to calculate a larger pruning bound (≥ 0). Therefore, the pruning power of DAP is stronger than the basic DA as illustrated in Figure 4, and consequently shows better time performance in Figure 5.

Note that the pruning rates of Rule 2 are already very high (greater than 0.9). Therefore, even by applying advanced techniques, we cannot achieve much better pruning bounds anymore. The corresponding time cost is then similar as well. Moreover, although the DAP approach shows quite similar pruning power to the basic DA in Rules 3 and 4, as we discussed in Section V, the DAP algorithm is at least no worse than DA.

Processing Order in $\mathcal{C}_Y$: Now, we discuss the processing orders of candidates in $\mathcal{C}_Y$, i.e., the mid-first order and the top-first order. Table V presents the time cost of different approaches with these two orders, in the Rule 1 test. Similar conclusions can also be observed in other rules.

TABLE V
TIME COST (S) OF VARIOUS PROCESSING ORDER IN $\mathcal{C}_Y$

l-th largest	Mid-first in PAP		Top-first in PAP	
	DA	DAP	DA	DAP
1	**34.20**	23.21	34.40	**18.50**
2	**37.60**	24.92	39.00	**20.40**
3	**39.93**	25.90	41.36	**21.15**
4	**41.26**	26.75	42.96	**21.93**
5	**42.23**	28.73	44.11	**23.90**
6	**44.14**	28.95	46.18	**24.53**
7	**44.93**	29.70	47.79	**25.06**

As discussed in Section V, when the pruning bound is initially set to $V_{\max} = 0$ in the DA+PAP approach, the mid-first order is preferred in order to find a large $V_{\max}$ as early as possible. Therefore, as presented in Table V, DA with mid-first order in PAP (column 2) has lower time cost than the top-first one (column 4), under all answer size l. On the other hand, if advanced pruning bound $V_{\max} > 0$ is calculated by DAP, then we can use the top-first order to prune more candidates in $\mathcal{C}_Y$. Consequently, the top-first order of PAP shows better performance than the mid-first one, in approaches together with DAP. Nevertheless, the DAP+PAP with top-first order can always achieve the lowest time cost, and the improvement of mid-first order in DA+PAP is not as significant as the DAP+PAP approaches as well. Thereby, we could use top-first order in most cases.

Scalability on l-th Largest: Finally, we also study the efficiency of determining l-largest expected utility answers over various data sizes. Here, we only report the result of 5-th largest $\bar{\mathrm{U}}$ answers in Figure 6. Similar conclusions are observed in other l sizes. Again, since there is no pruning of the DA+PA approach, the time cost of DA+PA in Figure 6 is

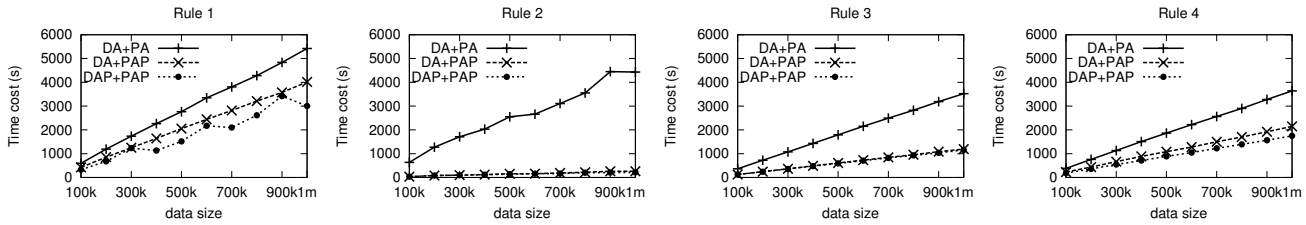

Fig. 6. Scalability on data sizes (return 5-th largest $\bar{U}$)

the same as Figure 2 of the 1st largest. The improvement of pruning like DA+PAP in Figure 6 is not as significant as that of 1st largest answers. For example, in Rule 1 with 1m data size, DA+PAP has an improvement about 2200 (from 5500 to 3300) in Figure 2, while the corresponding improvement is only about 1500 (from 5500 to 4000) in Figure 6. Nevertheless, DAP+PAP can still keep the lowest time cost in all the test in Figure 6.

VII. Conclusions

Motivated by the utility of differential dependencies (DDs) in real applications like violation detection, in this paper, we study the problem of determining the distance thresholds for metric distance constraints from data. Instead of manually specifying requirements of various statistical measures, we conduct the determination in a parameter-free style, i.e., to compute an expected utility of the distance threshold pattern and return the results with the maximum expected utility. Several advanced pruning algorithms are then developed in order to efficiently find the desired distance thresholds. Finally, the experimental evaluation on three real data sets demonstrates the performance of our proposed methods. In particular, we evaluate the effectiveness of returned results in the violation detection application, and show that the pruning techniques can achieve lower time cost.

Acknowledgments

The work described in this paper was partially supported by Hong Kong RGC GRF Project No. 611411; National Grand Fundamental Research 973 Program of China under Grant 2012CB316200; Hong Kong RGC GRF Project No. CUHK 411211; China NSFC Project No. 61073005.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Searching with numbers. In *WWW*, pages 420–431, 2002.

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

[4] 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.

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

[6] T. Calders, R. T. Ng, and J. Wijsen. Searching for dependencies at multiple abstraction levels. *ACM Trans. Database Syst.*, 27(3):229–260, 2002.

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

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

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

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

[11] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *ICDE*, pages 1231–1234, 2009.

[12] W. Fan, J. Li, X. Jia, and S. Ma. Reasoning about record matching rules. *PVLDB*, 2009.

[13] Full Version. Parameter-free determination of distance thresholds for metric distance constraints. *http://ise.thss.tsinghua.edu.cn/sxsong/doc/free.pdf*.

[14] 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.

[15] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a dbms for approximate string processing. *IEEE Data Eng. Bull.*, 24(4):28–34, 2001.

[16] J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, pages 211–218, 2002.

[17] 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.

[18] J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theor. Comput. Sci.*, 149(1):129–149, 1995.

[19] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian. Metric functional dependencies. In *ICDE*, pages 1275–1278, 2009.

[20] S. Kramer and B. Pfahringer. Efficient search for strong partial determinations. In *KDD*, pages 371–374, 1996.

[21] H. Mannila and K.-J. Räihä. Algorithms for inferring functional dependencies from relations. *Data Knowl. Eng.*, 12(1):83–99, 1994.

[22] 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.

[23] D. Montgomery and G. Runger. *Applied Statistics and Probability for Engineers*. Wiley, 2nd edition, 1999.

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

[25] T. Scheffer. Finding association rules that trade support optimally against confidence. *Intell. Data Anal.*, 9(4):381–395, 2005.

[26] J. C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *ICML*, pages 284–290, 1993.

[27] S. Song and L. Chen. Discovering matching dependencies. In *CIKM*, pages 1421–1424, 2009.

[28] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Trans. Database Syst.*, 36(3):16, 2011.

[29] S. Song, L. Chen, and P. S. Yu. On data dependencies in dataspaces. In *ICDE*, pages 470–481, 2011.

[30] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[31] G. I. Webb and S. Zhang. K-optimal rule discovery. *Data Min. Knowl. Discov.*, 10(1):39–79, 2005.