Efficient Determination of Distance Thresholds for Differential Dependencies

Shaoxu Song, Lei Chen, and Hong Cheng

Abstract—The importance of introducing distance constraints to data dependencies, such as differential dependencies (DDs), has recently been recognized. The differential dependencies are tolerant to small variations, which enable them to apply to wide data quality checking applications, such as detecting data violations. However, the determination of distance thresholds for the differential dependencies 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 differential dependencies, 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 use 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.

Index Terms—Database integration, heterogeneous databases

1 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 [2] for a survey). Recently, *functional dependencies* (FDs) have been revisited and revised with extensions [3] to capture the inconsistency in the dirty data [4]. 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.

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

The constraints are useful in detecting data violations, a very important task for data cleaning [5]. For instance, we can use the above fd_1 to detect violations in an instance of Hotel in Table 1. 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 1 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.

To address small variations in data formats, functional dependencies have recently been extended by incorporating distance constraints, namely *differential dependencies* (DDs) [1]. 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 [6] 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

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

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 sides of attributes. There are some other notations that specify distance metrics only in one side [7], [8] and could be regarded as special cases. Indeed, when all the thresholds

- *S. Song is with the Key Laboratory for Information System Security, Ministry of Education; TNList; School of Software, Tsinghua University. E-mail: sxsong@tsinghua.edu.cn.*
- *L. Chen is with the Hong Kong University of Science and Technology, Hong Kong, China. E-mail: leichen@cse.ust.hk.*
- *H. Cheng is with the Chinese University of Hong Kong, Hong Kong, China. E-mail: hcheng@se.cuhk.edu.hk.*

Manuscript received 24 Oct. 2012; revised 21 Mar. 2013; accepted 21 Apr. 2013. Date of publication 21 May 2013; date of current version 31 July 2014. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier 10.1109/TKDE.2013.84

TABLE 1
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

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 ([Address] $\rightarrow$ [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 differential dependencies. 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 [9], 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 differential dependencies 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 [10], we may also study the measures for DDs, in order to indicate how reliable and useful a differential dependency is. As opposed to the equality function in the previous work, the major difference of measures for differential dependencies is about the tolerance via distance metrics.

Specifically, the utility of differential dependencies could be investigated by certain statistical measures including *support*, *confidence* [11] 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 differential dependencies 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]

1. A large $\varphi[X]$ could be meaningful. See Section 3.1 for a discussion.

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 ([Address] $\rightarrow$ [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 1 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 differential dependencies 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 2 and the preliminary of this study in Section 3. Section 4 develops the computation of expected utility, together with analysis on its semantics. In Section 5, we present pruning algorithms. Our extensive experimental evaluation is reported in Section 6. Finally, Section 7 concludes this paper.

2 Related Work

The importance of similarity constraints has recently been recognized for data dependencies [12]–[14]. Besides the differential dependencies (DDs) [1], other notations of metric distance constraints are also studied.

Dependency with Metric: Koudas *et al*. [7] 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. As MFDs are tolerant of small variations in the Y attributes, they are more useful than the traditional FDs in detecting data violations. Moreover, the notation of differential dependencies (DDs) [1], introducing distance metrics on both sides of attributes, can have even more expressive power as illustrated in the introduction. Indeed, 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*. [8] 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. Reasoning mechanism for deducing MDs from a set of given MDs is studied in [8]. As illustrated, MDs and their reasoning techniques can improve the effectiveness of record matching. A sound and complete inference system is also presented in [15] for the deduction analysis of MDs. In addition, comparable dependencies (CDs) over heterogeneous data are also studied in [16], which need the more complicated schema mapping support. Golab *et al*. [17] propose sequential dependencies, which also target on the metric distance/similarity of Y values. A sequential dependency, in the form of $X \rightarrow_g Y$, states that when tuples are sorted on X, the distance/similarity between the Y-values of any two consecutive tuples should be within the interval g. We believe that our proposed threshold determination techniques could be useful and possibly extended for determining these more complex metric distance constraints in the future work.

Dependency Discovery: The DDs discovery in [1] targets on a minimal cover of all DDs that hold in a data instance, while the statistical measures for DDs w.r.t. the utility are not studied. Such statistical measures are essential to tell the importance of a DD. Therefore, in this study, we first introduce three statistical measures, i.e., support, confidence and dependent quality, for DDs, and then study the distance threshold determination regarding the statistical measures.

Indeed, the discovery of data dependencies from a data instance has been widely studied [18]–[20]. In discovering FDs, Huhtala *et al*. [9], [21] propose a level-wise algorithm, namely TANE, together with efficient pruning when searching in the lattice of attributes. Remarkably, TANE algorithm also supports the discovery of approximate FDs, with statistical measures. Wyss *et al*. [22] study a depth-first, heuristic-driven algorithm, namely FastFDs, which is (almost) linear to the size of FDs cover. Flach and Savnik [23] discover FDs in a bottom-up style, which considers the maximal invalid dependencies first. When searching in a hypotheses space, the maximum invalid dependencies are used for pruning the search space. Fan *et al*. [24] also extend the above algorithms for discovering FDs with conditions. 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 [25] about MDs discovery. It differs from our current work in two aspects: (1) the determination on metric distance only needs to be considered in the determinant side X for MDs; and (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.

Dependency Measure: A dependency rule can be measured in various ways. In the measures of FDs [9], g_3 measure [10] 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 [11] and conditional FDs [24], [26], [27]. 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 [11]. In our study, we also utilize support and confidence for differential dependencies, which are defined based on tuple pairs. The major difference of measures for differential dependencies to the previous work is about dependent quality. As introduced in the introduction, differential dependencies need an additional measure to evaluate the quality of tolerance in the dependent attributes.

Instead of setting minimum requirements of several measures, in association rule mining [28], it is also studied to return the most interesting rules according to the specified measures. For example, Han *et al*. [29] 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 [30] study the k-optimal rule discovery, where the leverage is used as a single value measure instead of support and confidence separately. Scheffer [31] 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 differential dependencies 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

TABLE 2
Notations

Symbol	Description
φ	Pattern of distance thresholds
$\mathrm{D}(\varphi)$	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

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 semantics as illustrated in Section 4.

3 Problem Statement

In this section, we first introduce the formal definition and statistical measures for differential dependencies. It raises the problem of determining thresholds with high utility. Table 2 lists the frequently used notations in this paper.

3.1 Preliminary

Consider a relation $\mathcal{R}$. For each attribute $A \in \mathcal{R}$, we associate a distance metric d_A, e.g., edit distance [32] or cosine similarity [33] on text, or the metric on numeric data [34]. The selection of distance metric is out of the scope of this study. Please refer to [6] for a survey. Let $d_A(a_1, a_2)$ denote the distance between two values a_1 and a_2 on attribute A.

Definition 1. *A* differential dependency *(DD) [1] is a pair $(X \rightarrow Y, \varphi)$, where (1) $X \subseteq \mathcal{R}$ denotes a set of* determinant attributes*; (2) $Y \subseteq \mathcal{R}$ is a set of* dependent attributes*; and (3) φ is a* pattern (vector) of distance thresholds *on attributes X and Y, also denoted by $\varphi[XY]$, where the projection $\varphi[A]$ of φ 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 $A \in \mathcal{R} \backslash (X \cup Y)$ 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.

3.2 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 [11], 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 re-computation among the evaluation of different φ, we can pre-compute 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.

Definition 2. *The* confidence *of φ is the ratio of tuple pairs satisfying $\varphi[XY]$ to the pairs satisfying $\varphi[X]$.*

$$\mathrm{C}(\varphi) = \frac{\mathsf{count}(b \vDash \varphi[XY])}{\mathsf{count}(b \vDash \varphi[X])}. \tag{1}$$

For example, in Table 1, 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.

Definition 3. *We define $\mathrm{D}(\varphi)$ as the support of $\varphi[X]$ (the left-hand-side of φ).*

$$\mathrm{D}(\varphi) = \frac{\mathsf{count}(b \vDash \varphi[X])}{M}. \tag{2}$$

To illustrate the computation of statistical measures, we consider the previously used running example in Table 1 in the Introduction. Since there are 15 tuple pairs in Table 1 and 6 of them have distance ≤ 8 on Address, we have $\mathrm{D}(\mathsf{dd}_1) = 6/15 = 0.4$.

Definition 4. *The overall* support *of φ is the proportion of tuple pairs whose distances satisfy the thresholds in $\varphi[XY]$ on both attributes of X and Y.*

$$\mathrm{S}(\varphi) = \mathrm{C}(\varphi)\mathrm{D}(\varphi). \tag{3}$$

Consequently, the support of dd_1 can be computed, i.e., $\mathrm{S}(\mathsf{dd}_1) = 4/15$.

Definition 5. *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.*

$$\mathrm{Q}(\varphi) = \frac{\sum_{A \in Y} d_{\max} - \varphi[A]}{|Y| \times d_{\max}}. \tag{4}$$

Intuitively, as presented in formula 4, 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.

3.3 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.

For any matching tuple b, a possible interpretation of the utility $\mathrm{U}(\varphi)$ of a dependency φ could be the prediction accuracy of b satisfying $\varphi[Y]$ with dependent quality $\mathrm{Q}(\varphi)$ given b satisfies $\varphi[X]$, i.e.,

$$\Pr(b \vDash \varphi[Y], \mathrm{Q}(\varphi) \mid b \vDash \varphi[X]), \tag{5}$$

Intuitively, according to this prediction, high confidence and high dependent quality may lead to high utility. However, the contribution of support in utility is not investigated yet. Therefore, we aim to refine the utility w.r.t. confidence and dependent quality by using support, i.e., to compute the *expected utility*,

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

where $\mathrm{C}(\varphi), \mathrm{D}(\varphi)$ and $\mathrm{Q}(\varphi)$ are the statistics observed from the matching relation $\mathcal{M}$. It is to consider the probabilities of possible utility values u. The computation and semantics of the expected utility are discussed in the following Section 4. 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 6. *Given the determinant attributes X and the dependent attributes Y, 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 $\bar{\mathrm{U}}(\varphi)$.*

4 Computing Expected Utility

In this section, we present the detailed computation steps of the expected utility, and then analyze the properties of expected utility to illustrate its semantics.

4.1 Computation

Given a data instance, we can compute the aforesaid statistics $\mathrm{C}(\varphi), \mathrm{D}(\varphi), \mathrm{Q}(\varphi)$ denoted by C, D, 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,

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

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 [35], 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., $n = \mathrm{D}$ (Recall that D denotes $\mathrm{D}(\varphi)$ for simplicity). Among them, we observed $\mathrm{D} \times \mathrm{C} \times \mathrm{Q}$ instances satisfying $\varphi[Y]$ with high dependent quality (i.e., $k = \mathrm{D} \times \mathrm{C} \times \mathrm{Q}$ successes). Moreover, according to formula 5, 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 $f(\mathrm{D} \times \mathrm{C} \times \mathrm{Q}; \mathrm{D}, u)$. Thereby, we can rewrite the computation formula,

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

$P(\mathrm{U} = u \mid \mathrm{D})$ 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 accuracy of utility u, i.e., the probability of matching tuples that satisfy $\varphi[X]$ also satisfying $\varphi[Y]$ with dependent quality $\mathrm{Q}(\varphi)$. Thus, we use the histogram $P(\mathrm{CQ})$ to estimate $P(\mathrm{U} = u)$. Let $\overline{\mathrm{CQ}}$ be the mean of CQ. Following the intuition of modeling prediction accuracy as binomial distribution [36], we also treating the prior of utility (prediction accuracy) in a similar way. Specifically, each matching tuple can be identified as either satisfying a φ or rejecting it. The chance of the former happening is $\overline{\mathrm{CQ}}$. It leads to a Binomial distribution, denoted by $\pi(u)$,

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

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

$$\bar{\mathrm{U}}(\varphi) = E(\mathrm{U} \mid \mathrm{C}, \mathrm{D}, \mathrm{Q}) = \frac{\int u f(\mathrm{D} \times \mathrm{C} \times \mathrm{Q}; \mathrm{D}, u) \pi(u) du}{\int f(\mathrm{D} \times \mathrm{C} \times \mathrm{Q}; \mathrm{D}, u) \pi(u) du}. \tag{6}$$

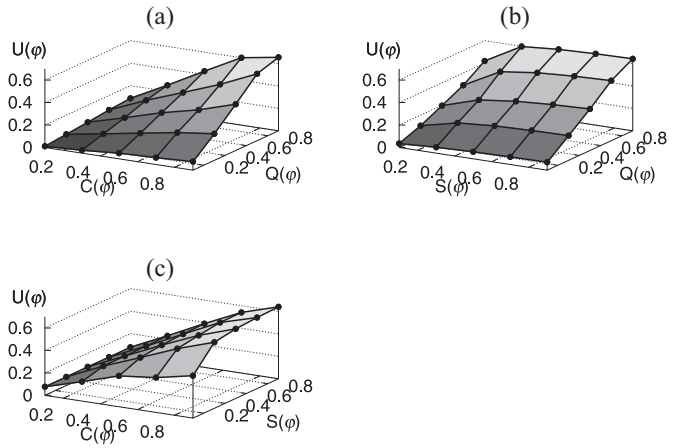


Fig. 1. Contribution to expected utility. (a) Same support $S(\varphi)$. (b) Same confidence $C(\varphi)$. (c) Same dependent quality $Q(\varphi)$.

4.2 Semantics

We discuss the semantics of the expected utility, that is, how the support, confidence and dependent quality measures of φ contribute to the expected utility $\bar{U}(\varphi)$.

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

Proof. First, according to $D(\varphi_1)C(\varphi_1) = \rho D(\varphi_2)C(\varphi_2)$ and $Q(\varphi_1) \geq \frac{1}{\rho}Q(\varphi_2)$, we have

$$\Delta = D(\varphi_1)C(\varphi_1)Q(\varphi_1) - D(\varphi_2)C(\varphi_2)Q(\varphi_2) \geq 0.$$

Moreover, since $C(\varphi_1) \geq \rho C(\varphi_2)$, we have

$$\Delta' = D(\varphi_2) - D(\varphi_1) \geq 0$$

as well. According to Lemma 3 in Supplemental Results, which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.84 we prove the conclusion. □

This conclusion verifies our intuition that higher support, confidence and dependent quality will contribute to a larger expected utility. In Fig. 1, we use running examples to illustrate the expected utility of different φ, which have various D, C, Q measures. To illustrate the contributions of measures clearly, we fix one of them, and observe the contributions of the other two measures. For instance, Proposition 1 states that if any φ_1, φ_2 have the same support, and the confidence and dependent quality of φ_1 are higher than those of φ_2, then the utility $\bar{U}(\varphi_1)$ is larger than that of φ_2 as well. As presented in Fig. 1(a), under the same support $S(\varphi) = 0.8$, with the increase of $C(\varphi)$ and $Q(\varphi)$, the corresponding expected utility $\bar{U}(\varphi)$ is also larger. Although the confidence of a φ could be high, e.g., $C(\varphi) = 0.9$, if the corresponding dependent quality is low, e.g., $Q(\varphi) = 0.1$, the expected utility $\bar{U}(\varphi) = 0.07$ is still low. Similar conclusions can be drawn under the same confidence $C(\varphi)$ or dependent quality $Q(\varphi)$ in Fig. 1(b) and (c), according to Proposition 2 and Proposition 3 respectively.

Proposition 1. *For any φ_1, φ_2 with the same support*

$$S(\varphi_1) = S(\varphi_2),$$

if the confidence and dependent quality of φ_1 are higher than those of φ_2 respectively, i.e., $C(\varphi_1) \geq C(\varphi_2)$, and $Q(\varphi_1) \geq Q(\varphi_2)$, we have $\bar{U}(\varphi_1) \geq \bar{U}(\varphi_2)$.

Proposition 2. *For any φ_1, φ_2 with the same confidence*

$$C(\varphi_1) = C(\varphi_2) = C,$$

if φ_1 has higher support than φ_2, $S(\varphi_1) = \rho S(\varphi_2), \rho \geq 1$, and the dependent quality of φ_1 is higher than that of φ_2 as follows $Q(\varphi_1) \geq \frac{1}{\rho}Q(\varphi_2) + \frac{\rho-1}{C}$, then we have $\bar{U}(\varphi_1) \geq \bar{U}(\varphi_2)$.

Proposition 3. *For any φ_1, φ_2 with the same dependent quality*

$$Q(\varphi_1) = Q(\varphi_2),$$

if φ_1 has higher support than φ_2, $S(\varphi_1) = \rho S(\varphi_2), \rho \geq 1$, and the confidence of φ_1 is higher than that of φ_2 as follows $C(\varphi_1) \geq \rho C(\varphi_2)$, then we have $\bar{U}(\varphi_1) \geq \bar{U}(\varphi_2)$.

Referring to formula (6) of computing $\bar{U}(\varphi)$, Proposition 1 and Proposition 3 can be directly proved by Lemma 3, and Proposition 2 is proved by Lemma 4 in Supplemental Results, available online.

So far, we have presented the computation of expected utility with clarification on the corresponding semantics. 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.

5 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., $\varphi_{\max} = \arg\max_{\varphi} \bar{U}(\varphi)$. 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 $\bar{U}(\varphi)$.

The previous FDs discovery considers the combination of attributes [9] and targets on minimizing a single measure, such as g_3 [10]. 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.

5.1 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 $\bar{U}(\varphi)$.

For an attribute $A \in Y$, we can consider the search space of distance threshold $\varphi[A]$ from 0 to $d_{\max}$. Let $\mathcal{C}_Y$ denote the set of distance threshold patterns $\varphi[Y]$ by enumerating all the distance thresholds $\varphi[A] \in$ dis for all the dependent attributes $A \in Y$.

Since $\varphi[X]$ on the determinant attributes X is fixed in the current step, according to formula 2, the $D(\varphi)$ value is the same for any φ such that $\varphi[Y] \in \mathcal{C}_Y$. Therefore, we mainly study the other two measures $C(\varphi)$ and $Q(\varphi)$ in terms of contributions to the expected utility $\bar{U}(\varphi)$.

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

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

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

Proof. Let

$$\Delta = \frac{\text{C}(\varphi_1)\text{Q}(\varphi_1) - \text{C}(\varphi_2)\text{Q}(\varphi_2)}{\text{D}} \geq 0$$

and

$$\Delta' = \text{D}(\varphi_2) - \text{D}(\varphi_1) = 0.$$

According to Lemma 3 in Supplemental Results, available online, we can prove the conclusion. □

According to Theorem 2, for a fixed $\varphi[X]$, to find a φ with the maximum $\bar{\text{U}}(\varphi)$ is equivalent to find the one with the maximum $\text{C}(\varphi)\text{Q}(\varphi)$. For each $\varphi[Y] \in \mathcal{C}_Y$, we can directly calculate the dependent quality $\text{Q}(\varphi)$ by formula 4. Moreover, together with the given $\varphi[X]$ on the determinant attributes X, we can also compute the measure $\text{C}(\varphi)$ from $\mathcal{M}$ using formula 1. The result φ with the maximum $\bar{\text{U}}(\varphi)$ 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 $\bar{\text{U}}(\varphi)$ is maximized. As mentioned, to find the result, we only need to compute $\text{C}(\varphi_i)\text{Q}(\varphi_i)$ 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 $\text{C}(\varphi_i)$ from $\mathcal{M}$ by using formula 1. Let $V_{\max}$ denote the maximum value of $\text{C}(\varphi)\text{Q}(\varphi)$ in the first $i-1$ candidates, i.e.,

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

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

Example 1. Supposing that the maximum distance value is 10, we denote all the distance thresholds as $\mathsf{dis} = \{0, 1, 2, \ldots, 10\}$. Consider two attributes in Y, e.g., "venue" and "year" in Rule 1 used in the experiments in Section 6, having the same $\mathsf{dis} = \{0, 1, 2, \ldots, 10\}$. Fig. 2(a) illustrates all the distance threshold patterns in $\mathcal{C}_Y$, from $< 0, 0 >$ to $< 10, 10 >$. Each node, such as $< 1, 2 >$, corresponds to a pattern $\varphi[Y] \in \mathcal{C}_Y$. That is, the distance threshold of the first attribute "venue" is 1, and the second attribute "year" has distance threshold equal to 2. Algorithm 1 traverses all these patterns in Fig. 2(a) to find a $\varphi[Y]$, which has the maximum $\text{C}(\varphi)\text{Q}(\varphi)$ value. For instance, according to the results in Table 3 in the experiments, the pattern $< 3, 1 >$ for Y attributes "venue" and "year" has the maximum $\text{C}(\varphi)\text{Q}(\varphi)$ values, i.e., 0.376*0.8. As shown in Table 3, such a result with a high expected utility (large confidence $\text{C}(\varphi)$ and dependent quality $\text{Q}(\varphi)$) may leads to relatively good performance in applications (high precision and recall).

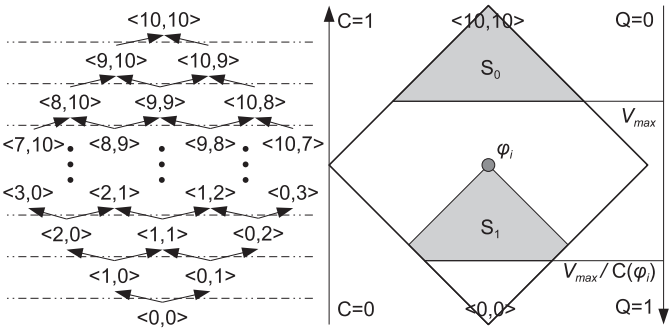


Fig. 2. Pruning on dependent attributes. (a) Dominant relationships. (b) Measure relationships.

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 $\text{C}(\varphi)$. According to formula 1, 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.

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 7. *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 $\varphi_1[Z] \prec \varphi_2[Z]$.*

Fig. 2(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., $\varphi_1[Y] \prec \varphi_2[Y]$. 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 $\varphi_1[Y] \prec \varphi_2[Y]$ and $\varphi_2[Y] \prec \varphi_3[Y]$, we have $\varphi_1[Y] \prec \varphi_3[Y]$ as well. For example, in Fig. 2(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 $\varphi_1[Y] \prec \varphi_2[Y]$, then $\text{C}(\varphi_1) \geq \text{C}(\varphi_2)$ and $\text{Q}(\varphi_1) \leq \text{Q}(\varphi_2)$.*

Proof. Let b be any matching tuple from $\mathcal{M}$. First of all, since $\varphi_1[X] = \varphi_2[X]$, we have

$$\mathsf{count}(b \vDash \varphi_1[X]) = \mathsf{count}(b \vDash \varphi_2[X])$$

as well. Moreover, for any b satisfying φ_2, having $b[A] \leq \varphi_2[A], A \in Y$, according to Definition 7, we always have $b[A] \leq \varphi_2[A] \leq \varphi_1[A]$, i.e., b also satisfies φ_1. In other words, $\mathsf{count}(b \vDash \varphi_1[XY])$ is greater than that of φ_2, that is, $\text{C}(\varphi_1) \geq \text{C}(\varphi_2)$.

Next, since we have $\varphi_1[A] \geq \varphi_2[A]$ for each attribute $A \in Y$, the dependent quality $\text{Q}(\varphi_1)$ is lower than $\text{Q}(\varphi_2)$. □

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 Fig. 2(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 4. *For any* $\varphi_k[Y] \in \mathcal{C}_Y$ *with* $\mathrm{Q}(\varphi_k) \le V_{\max}$, *we have* $\bar{\mathrm{U}}(\varphi_k) \le \bar{\mathrm{U}}(\varphi_{\max})$.

Proof. Since the confidence is always less than or equal to 1, we have $\mathrm{C}(\varphi_k)\mathrm{Q}(\varphi_k) \le \mathrm{Q}(\varphi_k) \le V_{\max} = \mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max})$. According to Theorem 2, we have $\bar{\mathrm{U}}(\varphi_k) \le \bar{\mathrm{U}}(\varphi_{\max})$. □

Let $S_0 = \{\varphi_k \mid \mathrm{Q}(\varphi_k) \le V_{\max}, \varphi_k[Y] \in \mathcal{C}_Y\}$. Then, 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 5. *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* $\bar{\mathrm{U}}(\varphi_k) \le \bar{\mathrm{U}}(\varphi_{\max})$.

Proof. According to $\varphi_i[Y] \lessdot \varphi_k[Y]$, we have $\mathrm{C}(\varphi_k) \le \mathrm{C}(\varphi_i)$. It implies $\mathrm{C}(\varphi_k)\mathrm{Q}(\varphi_k) \le \mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_k) \le V_{\max} = \mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max})$. Referring to Theorem 2, the conclusion is proved. □

Let $S_1 = \{\varphi_k \mid \varphi_i \lessdot \varphi_k, \mathrm{Q}(\varphi_k) \le \frac{V_{\max}}{\mathrm{C}(\varphi_i)}, \varphi_k[Y] \in \mathcal{C}_Y\}$. According to Proposition 5, all the φ_k in S_1 can be safely pruned as well, without computing $\mathrm{C}(\varphi_k)$.

Example 2. Let φ_i in Fig. 2(b) denote the current candidate, and $V_{\max}$ be the maximum value of $\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$ in the first $i-1$ candidates. As mentioned, the dependent quality increases with a downward traversal of candidates in the dominant graph. The shaded area S_0 denotes the candidates φ_k with $\mathrm{Q}(\varphi_k) \le V_{\max}$. According to Proposition 4, all the candidates in S_0 are safely pruned. Similarly, according to Proposition 5, those candidates φ_k with $\mathrm{Q}(\varphi_k) \le \frac{V_{\max}}{\mathrm{C}(\varphi_i)}$ and dominated by φ_i are pruned as well, i.e., the other shaded area of S_1 in Fig. 2(b).

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 5. 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 4.

Algorithm 2 Determination of Thresholds 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** $\varphi_{\max}$

Algorithm 3 Determination of Thresholds 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** $\varphi_{\max}$

5.2 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 2. 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 6 as presented in Section 4. 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 $U_{\max}$.

Example 3. As $\mathcal{C}_Y$ in Fig. 2(a), we can also enumerate all the distance threshold patterns for $\mathcal{C}_X$. Let the bar in Fig. 3(a) denote the sequence of all $\varphi_i[X] \in \mathcal{C}_X$. Algorithm 3 traverses all these candidates in $\mathcal{C}_X$ to find a $\varphi_{\max}[X]$ which has the maximum expected utility. When computing the dependents for each $\varphi_i[X]$, the initial pruning bound can be simply given as $V_{\max} = 0$.

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

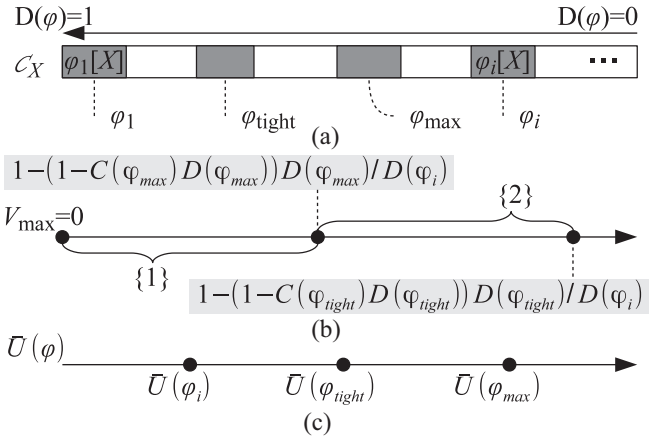


Fig. 3. Pruning on determinant attributes. (a) Candidate order. (b) Pruning bound. (c) Expected utility.

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

Proof. Let

$$\Delta = \mathrm{D}(\varphi_1)\mathrm{C}(\varphi_1)\mathrm{Q}(\varphi_1) - \mathrm{D}(\varphi_2)\mathrm{C}(\varphi_2)\mathrm{Q}(\varphi_2)$$

and

$$\Delta' = \mathrm{D}(\varphi_1) - \mathrm{D}(\varphi_2) \geq 0.$$

With the condition of $\mathrm{C}(\varphi_1)$ and $\mathrm{C}(\varphi_2)$, we have $\Delta \geq \Delta'$. According to Lemma 4 in Supplemental Results, available online, we can prove the conclusion. □

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

$$V_{\max} = 1 - \frac{\mathrm{D}(\varphi_{\max})}{\mathrm{D}(\varphi_i)}\Big(1 - \mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max})\Big) \qquad (7)$$

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

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: $V_{\max} = 1 - \frac{\mathrm{D}(\varphi_{\max})}{\mathrm{D}(\varphi_i)}\Big(1 - \mathrm{C}(\varphi_{\max})\mathrm{Q}(\varphi_{\max})\Big)$
5: compute $\varphi_i[Y]$ by PAP($\mathcal{M}, \varphi_i[X], V_{\max}$)
6: **if** $\varphi_i[Y]$ exists **then**
7: $\bar{\mathrm{U}}(\varphi_i) = E(\mathrm{U}(\varphi_i) \mid \mathrm{C}(\varphi_i), \mathrm{D}(\varphi_i), \mathrm{Q}(\varphi_i))$
8: **if** $U_{\max} < \bar{\mathrm{U}}(\varphi_i)$ **then**
9: $\varphi_{\max} = \varphi_i, U_{\max} = \bar{\mathrm{U}}(\varphi_i)$
10: **return** $\varphi_{\max}$

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 7, 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 $\bar{\mathrm{U}}(\varphi_i)$.

Example 4. As shown in Fig. 3(a), candidates in $\mathcal{C}_X$ are ordered by $\mathrm{D}(\varphi)$. For any candidate φ_i with $\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$ less than the pruning bound $V_{\max}$ specified in formula (7), i.e., in the range of {1} in Fig. 3(b), it can be directly pruned according to Theorem 3, that is, with $\bar{\mathrm{U}}(\varphi_i) \leq \bar{\mathrm{U}}(\varphi_{\max})$ as illustrated in Fig. 3(c).

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 6 also verify that DAP+PAP is at least no worse than DA+PAP.

Tighter pruning bound using φ_j: Motivated by the importance of pruning bound, it is natural to clarify whether the pruning bound $V_{\max}$ developed by measures of $\varphi_{\max}$ in formula 7 is the tightest (largest) one. Indeed, $\varphi_{\max}$ only maximizes the $\bar{\mathrm{U}}(\varphi)$, but has no guarantee that the pruning bound $V_{\max}$ is maximized.

Consider the current φ_i. Recall that for a $j \in [1, i-1]$ we have $\mathrm{D}(\varphi_j) \geq \mathrm{D}(\varphi_i)$ according to the ordering of candidates in $\mathcal{C}_X$. Suppose that φ_i can also satisfy the other condition of $\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$ in Theorem 3, i.e., having $\bar{\mathrm{U}}(\varphi_j) \geq \bar{\mathrm{U}}(\varphi_i)$. For the result $\varphi_{\max}$ with the maximum expected utility in the previous $i-1$ steps, we have $\bar{\mathrm{U}}(\varphi_{\max}) \geq \bar{\mathrm{U}}(\varphi_j)$. Thereby, the pruning of φ_i by using φ_j is safe, according to

$$\bar{\mathrm{U}}(\varphi_{\max}) \geq \bar{\mathrm{U}}(\varphi_j) \geq \bar{\mathrm{U}}(\varphi_i).$$

In other words, any φ_j can be utilized in Theorem 3 to compute a pruning bound.

Algorithm 5 Determination of Thresholds for Determinant Attributes with Pruning Tight DAPT($\mathcal{M}$)

1: $U_{\max} = 0, W_{\min} = +\infty$
2: **for** each $\varphi_i[X] \in \mathcal{C}_X$ **do**
3: compute $\mathrm{D}(\varphi_i)$ from $\mathcal{M}$
4: $V_{\max} = 1 - \frac{W_{\min}}{\mathrm{D}(\varphi_i)}$
5: compute $\varphi_i[Y]$ by PAP($\mathcal{M}, \varphi_i[X], V_{\max}$)
6: **if** $\varphi_i[Y]$ exists **then**
7: $\bar{\mathrm{U}}(\varphi_i) = E(\mathrm{U}(\varphi_i) \mid \mathrm{C}(\varphi_i), \mathrm{D}(\varphi_i), \mathrm{Q}(\varphi_i))$
8: **if** $U_{\max} < \bar{\mathrm{U}}(\varphi_i)$ **then**
9: $\varphi_{\max} = \varphi_i, U_{\max} = \bar{\mathrm{U}}(\varphi_i)$
10: **if** $W_{\min} > \mathrm{D}(\varphi_i) - \mathrm{D}(\varphi_i)\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$ **then**
11: $W_{\min} = \mathrm{D}(\varphi_i) - \mathrm{D}(\varphi_i)\mathrm{C}(\varphi_i)\mathrm{Q}(\varphi_i)$
12: **return** $\varphi_{\max}$

Specifically, for all the φ_j in the previous $i-1$ steps, let

$$W_{\min} = \min_{j=1}^{i-1} \mathrm{D}(\varphi_j) - \mathrm{D}(\varphi_j)\mathrm{C}(\varphi_j)\mathrm{Q}(\varphi_j). \quad (8)$$

Then, the tightest pruning bound, i.e., the maximum $V_{\max}$, is

$$V_{\max} = 1 - \tfrac{W_{\min}}{\mathrm{D}(\varphi_i)} = \max_{j=1}^{i-1} 1 - \tfrac{\mathrm{D}(\varphi_j)}{\mathrm{D}(\varphi_i)}\Big(1 - \mathrm{C}(\varphi_j)\mathrm{Q}(\varphi_j)\Big).$$

Algorithm 5, namely DAPT, presents the implementation with the tighter pruning bound. Instead of computing the pruning bound $V_{\max}$ by using the current $\varphi_{\max}$ in the previous $i-1$ steps, the tighter bound is obtained by $W_{\min}$ in Line 4. According to formula 8, $W_{\min}$ denotes the minimum value of $\mathrm{D}(\varphi) - \mathrm{D}(\varphi)\mathrm{C}(\varphi)\mathrm{Q}(\varphi)$ for all the φ considered in the previous $i-1$ steps, which is computed in Line 11. Initially, we have $W_{\min} = +\infty$.

Example 5. As shown in Fig. 3(b), an even larger (tighter) pruning bound $V_{\max}$ can be obtained by formula (8) w.r.t. to a φ_{tight} that is previously processed before φ_i in Fig. 3(a). Consequently, those candidates in the range of {2} in Fig. 3(b) can be pruned, i.e., with $\bar{\mathrm{U}}(\varphi_i) \leq \bar{\mathrm{U}}(\varphi_{tight})$ as illustrated in Fig. 3(c). It is notable that these candidates cannot be identified with expected utility lower than $\bar{\mathrm{U}}(\varphi_{\max})$ by DAP in Algorithm 4 until $\bar{\mathrm{U}}(\varphi_i)$ is computed.

Note that the current $\varphi_{\max}$ is one of the first $i-1$ candidates. In other words, the above bound $V_{\max}$ is at least no smaller than the one calculated by using $\varphi_{\max}$ in formula 7 for DAP. Therefore, the worst case of DAPT algorithm will be the DAP.

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.

6 Experimental Evaluation

In the experimental evaluation, we use three real data sets. The *Cora*[2] data set, prepared by McCallum *et al*. [37], 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 [38] 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 $\mathcal{M}$.

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

*Rule*1 : *cora*(author, title → venue, year)
*Rule*2 : *cora*(venue → address, publisher, editor)
*Rule*3 : *restaurant*(name, address → city, type)
*Rule*4 : *citeseer*(address, affiliation, description → subject)

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

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.

6.1 Result Study in Real Applications

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 [39]. 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* $= \frac{|\text{truth} \cap \text{found}|}{|\text{found}|}$, *recall* $= \frac{|\text{truth} \cap \text{found}|}{|\text{truth}|}$, 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 3–6 illustrate the example results on Rules 1–4, 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 $\mathrm{S}(\varphi)$, confidence $\mathrm{C}(\varphi)$, dependent quality $\mathrm{Q}(\varphi)$, as well as the expected utility $\bar{\mathrm{U}}(\varphi)$. The results φ_i are listed in the descending order of $\bar{\mathrm{U}}(\varphi)$. In the last row, we also report the corresponding FDs, where the distance threshold is 0 for each attribute.

The results in Table 5 show the interesting case of independence. The distance thresholds of attributes name and type are 10 in all the results, which are 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

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 3
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 4
Effectiveness of Example Results from Rule 2

	$\varphi[X]$	$\varphi[Y]$			Measures				Violation Detection		
	venue	address	publisher	editor	$S(\varphi)$	$C(\varphi)$	$Q(\varphi)$	$\bar{U}(\varphi)$	Precision	Recall	F-measure
φ_1	4	10	1	10	0.1442	0.1241	0.30	0.0410	0.5043	0.1274	0.2034
φ_2	5	10	3	10	0.1742	0.1308	0.23	0.0362	0.5046	0.1528	0.2345
φ_3	1	9	5	10	0.0609	0.0558	0.20	0.0351	0.4991	0.0574	0.1029
φ_4	1	4	4	10	0.0609	0.0279	0.40	0.0351	0.5016	0.0594	0.1062
φ_5	1	5	3	10	0.0609	0.0279	0.40	0.0351	0.5016	0.0594	0.1062
φ_6	1	6	2	10	0.0609	0.0279	0.40	0.0351	0.5016	0.0594	0.1062
fd	0	0	0	0	0.0001	0.0014	1.00	0.0418	0.5283	0.0242	0.04628

TABLE 5
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

TABLE 6
Effectiveness of Example Results from Rule 4

	$\varphi[X]$			$\varphi[Y]$	Measures				Violation Detection		
	address	affiliation	description	subject	$S(\varphi)$	$C(\varphi)$	$Q(\varphi)$	$\bar{U}(\varphi)$	Precision	Recall	F-measure
φ_1	10	10	9	9	0.9918	0.4380	0.1	0.0314	0.5008	0.5582	0.5279
φ_2	10	10	5	9	0.5443	0.4732	0.1	0.0312	0.4945	0.2836	0.3604
φ_3	10	9	6	9	0.3590	0.4654	0.1	0.0259	0.5044	0.1936	0.2798
φ_4	10	9	7	9	0.3890	0.4611	0.1	0.0249	0.5023	0.2106	0.2967
φ_5	10	8	6	9	0.2703	0.4691	0.1	0.0226	0.4968	0.1426	0.2216
φ_6	10	7	10	9	0.1867	0.4675	0.1	0.0192	0.4859	0.0966	0.1611
fd	0	0	0	0	0.0000	0.0000	1.0	0.0187	–	0.0000	–

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 5 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 semantics analysis of expected utility. According to Theorem 1, if a φ (e.g., φ_2 in Table 3) has higher support, confidence and dependent quality than another (e.g., φ_4 in Table 3) at the same time, then the expected utility of φ_2 must be higher than that of φ_4. As observed in Table 3, we have $\bar{U}(\varphi_2) > \bar{U}(\varphi_4)$, which verifies our semantic analysis in Section 4.2. 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 3–6, 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

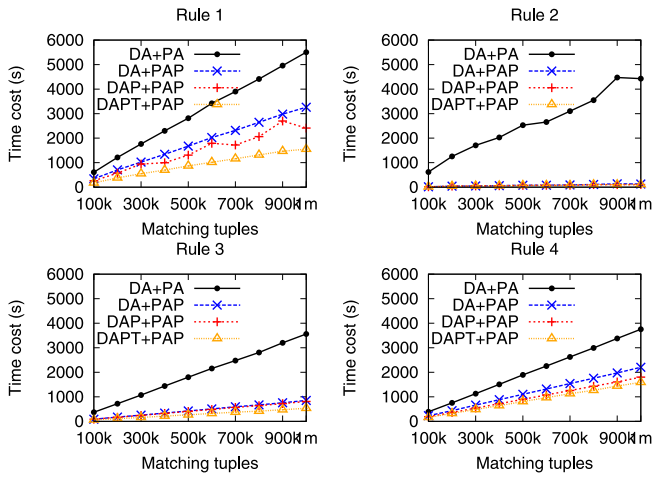


Fig. 4. Time performance on various data sizes (return largest $\bar{U}$).

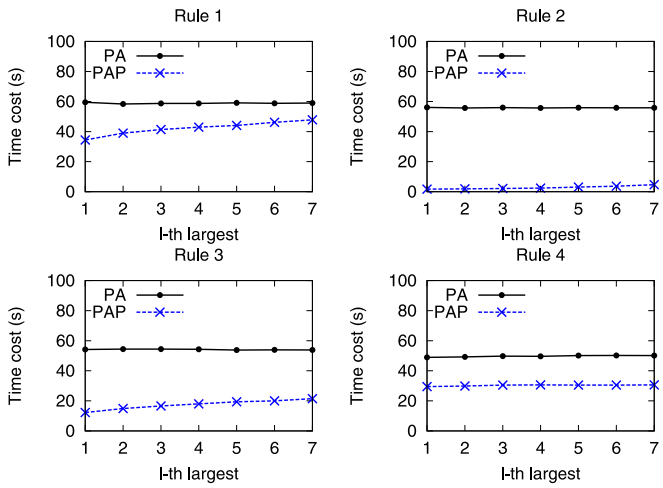


Fig. 5. Generation for dependent attributes.

of differential dependencies in the application. Although some of the measures of a φ are higher, e.g., φ_6 in Table 3 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.

Note that some results, e.g., φ_2 in Table 3, may have a better F-measure but lower expected utility. There are mainly two reasons for such cases. First, the expected utility provides a good guideline of choosing relatively useful dependency rules, but does not theoretically guarantee the exact accuracy in application. Second, the discovery of distance thresholds and the application of violation detection are conducted on two separated data parts. The gap between training (determination) and testing (detection) data may affect the performance as well. Nevertheless, in most cases, the dependency results with higher expected utilities yield better accuracy performance in the application.

As illustrated in Table 4, the utility of Rule 2 is extremely low. The rationale is that based on the only attribute "venue", it can barely determine the attributes of "address", "publisher" and "editor".

Finally, as shown in Tables 3–6, 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. In particular, for Rule 4 in Table 6, it is notable that almost no records would share the exactly same "description" and "subject" values. The functional dependency, defined on equality function, cannot address anything with support equal to 0, and consequently no violation is detected at all.

6.2 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, and the DAPT+PAP with tighter bounds. 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 6.3 below for the additional results on evaluating the individual methods, such as PA vs. PAP.

Fig. 4 reports the time cost of returning answers with the largest $\bar{U}$. 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 5.

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. **iii)** The DAPT+PAP approach with the tightest pruning bounds can always achieve the lowest time costs in all the rules under various data sizes. These results demonstrate the efficiency of the proposed pruning techniques, especially the superiority of the advanced DAPT+PAP approach.

6.3 Evaluation on Specific Steps

In this experiment, we demonstrate the detailed results by applying different methods in specific steps. As mentioned in Section 5, our algorithms can be easily extended to find the l-largest expected utility answers, which offer

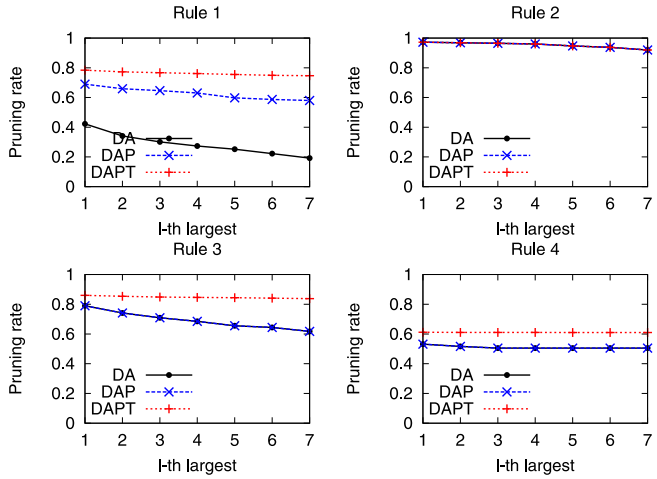


Fig. 6. Pruning power evaluation.

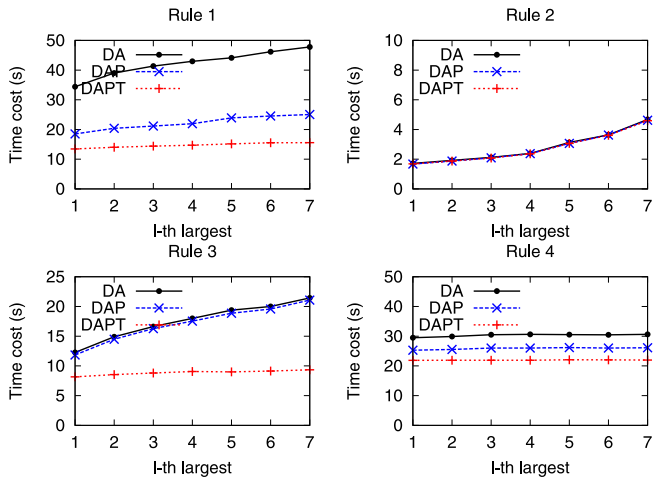


Fig. 7. Generation for determinant attributes.

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 Fig. 5. The basic PA algorithm has to traverse all the candidates on Y without any pruning. Therefore, as shown in Fig. 5, 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 4 and 5. 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 Fig. 5, time cost of PAP increases with a larger answer size l.

Moreover, the pruning power of PAP is affected by confidence and dependent quality measures, i.e., affected by data distribution. Thereby, 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 Fig. 5, PAP can achieve a significant improvement in Rule 2. Indeed, 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 three algorithms for determinant attributes, DA, DAP and DAPT, contribute different pruning bounds. In order to evaluate the pruning power of these approaches, Fig. 6 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, DAP and DAPT algorithms for determinant attributes X in Fig. 7.

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, and consequently shows better time performance in Fig. 7.

Moreover, the DAPT approach can introduce an even tighter pruning bound $V_{\max}$, by considering all the currently processed $i-1$ candidates. Note that the current $\varphi_{\max}$ is one of the first $i-1$ candidates. In other words, the pruning bound of DAPT is at least no worse than that of DAP. Indeed, as presented in Fig. 7, the DAPT algorithm can improve the time performance in most cases.

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 5, the DAP algorithm is at least no worse than DA. Meanwhile, the DAPT approach can achieve a better pruning rate in these two rules, and thus the corresponding time cost is lower.

7 Conclusion

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 differential dependencies 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 technique with tight bound can always achieve the lowest time cost.

Acknowledgments

The work described in this paper was partially supported by China NSFC Project No. 61202008 and 61370055; Hong Kong RGC GRF Project No. 611411, 411310 and 411211;

National Grand Fundamental Research 973 Program of China under Grant 2012CB316200; Microsoft Research Asia Grant and Google Faculty Award

REFERENCES

[1] S. Song and L. Chen, "Differential dependencies: Reasoning and discovery," *ACM TODS*, vol. 36, no. 3, Article 16, 2011.

[2] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Berlin, Germany: Springer, 2006.

[3] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis, "Conditional functional dependencies for data cleaning," in *Proc. IEEE 23rd ICDE*, Istanbul, Turkey, 2007, pp. 746–755.

[4] W. Fan, "Dependencies revisited for improving data quality," in *Proc. 27th PODS*, Vancouver, BC, Canada, 2008, pp. 159–170.

[5] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma, "Improving data quality: Consistency and accuracy," in *Proc. 33rd Int. Conf. VLDB*, Vienna, Austria, 2007, pp. 315–326.

[6] M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "Adaptive name matching in information integration," *IEEE Intell. Syst.*, vol. 18, no. 5, pp. 16–23, Sept. 2003.

[7] N. Koudas, A. Saha, D. Srivastava, and S. Venkatasubramanian, "Metric functional dependencies," in *Proc. IEEE 25th ICDE*, Shanghai, China, 2009, pp. 1275–1278.

[8] W. Fan, J. Li, X. Jia, and S. Ma, "Reasoning about record matching rules," *PVLDB*, vol. 2, no. 1, pp. 407–418, 2009.

[9] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *Comput. J.*, vol. 42, no. 2, pp. 100–111, 1999.

[10] J. Kivinen and H. Mannila, "Approximate inference of functional dependencies from relations," *Theor. Comput. Sci.*, vol. 149, no. 1, pp. 129–149, 1995.

[11] T. Calders, R. T. Ng, and J. Wijsen, "Searching for dependencies at multiple abstraction levels," *ACM TODS*, vol. 27, no. 3, pp. 229–260, 2002.

[12] R. Bassée and J. Wijsen, "Neighborhood dependencies for prediction," in *Proc. 5th PAKDD*, Hong Kong, China, 2001, pp. 562–567.

[13] F. Korn, S. Muthukrishnan, and Y. Zhu, "Checks and balances: Monitoring data quality problems in network traffic databases," in *Proc. 29th Int. Conf. VLDB*, Berlin, Germany, 2003, pp. 536–547.

[14] R. Belohlávek and V. Vychodil, "Data tables with similarity relations: Functional dependencies, complete rules and non-redundant bases," in *Proc. DASFAA*, Singapore, 2006, pp. 644–658.

[15] W. Fan, H. Gao, X. Jia, J. Li, and S. Ma, "Dynamic constraints for record matching," *VLDB J.*, Vol. 20, no. 4, pp. 1–26, 2010.

[16] S. Song, L. Chen, and P. S. Yu, "On data dependencies in dataspaces," in *Proc. IEEE 27th ICDE*, Hannover, Germany, 2011, pp. 470–481.

[17] L. Golab, H. J. Karloff, F. Korn, A. Saha, and D. Srivastava, "Sequential dependencies," *PVLDB*, vol. 2, no. 1, pp. 574–585, 2009.

[18] H. Mannila and K.-J. Räihä, "Algorithms for inferring functional dependencies from relations," *Data Knowl. Eng.*, vol. 12, no. 1, pp. 83–99, 1994.

[19] J. C. Schlimmer, "Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning," in *Proc. ICML*, Amherst, MA, USA, 1993, pp. 284–290.

[20] S. Kramer and B. Pfahringer, "Efficient search for strong partial determinations," in *Proc. KDD*, 1996, pp. 371–374.

[21] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen, "Efficient discovery of functional and approximate dependencies using partitions," in *Proc. 14th ICDE*, Orlando, FL, USA, 1998, pp. 392–401.

[22] C. M. Wyss, C. Giannella, and E. L. Robertson, "FastFDs: A heuristic-driven, depth-first algorithm for mining functional dependencies from relation instances - extended abstract," in *Proc. DaWaK*, Münich, Germany, 2001, pp. 101–110.

[23] P. A. Flach and I. Savnik, "Database dependency discovery: A machine learning approach," *AI Commun.*, vol. 12, no. 3, pp. 139–160, 1999.

[24] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong, "Discovering conditional functional dependencies," in *Proc. IEEE ICDE*, Shanghai, China, 2009, pp. 1231–1234.

[25] S. Song and L. Chen, "Discovering matching dependencies," in *Proc. 18th CIKM*, New York, NY, USA, 2009, pp. 1421–1424.

[26] L. Golab, H. J. Karloff, F. Korn, D. Srivastava, and B. Yu, "On generating near-optimal tableaux for conditional functional dependencies," *PVLDB*, vol. 1, no. 1, pp. 376–390, 2008.

[27] F. Chiang and R. J. Miller, "Discovering data quality rules," *PVLDB*, vol. 1, no. 1, pp. 1166–1177, 2008.

[28] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proc. SIGMOD*, Washington, DC, USA, 1993, pp. 207–216.

[29] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining top-k frequent closed patterns without minimum support," in *Proc. IEEE ICDM*, Washington, DC, USA, 2002, pp. 211–218.

[30] G. I. Webb and S. Zhang, "K-optimal rule discovery," *Data Min. Knowl. Discov.*, vol. 10, no. 1, pp. 39–79, 2005.

[31] T. Scheffer, "Finding association rules that trade support optimally against confidence," *Intell. Data Anal.*, vol. 9, no. 4, pp. 381–395, 2005.

[32] G. Navarro, "A guided tour to approximate string matching," *ACM CSUR*, vol. 33, no. 1, pp. 31–88, 2001.

[33] W. W. Cohen, "Integration of heterogeneous databases without common domains using queries based on textual similarity," in *Proc. SIGMOD*, Washington, DC, USA, 1998, pp. 201–212.

[34] R. Agrawal and R. Srikant, "Searching with numbers," in *Proc. 11th Int. Conf. WWW*, New York, NY, USA, 2002, pp. 420–431.

[35] D. Montgomery and G. Runger, *Applied Statistics and Probability for Engineers*, 2nd ed. New York, NY, USA: Wiley, 1999.

[36] T. Scheffer, "Average-case analysis of classification algorithms for Boolean functions and decision trees," in *Proc. ALT*, Sydney, NSW, Australia, 2000, pp. 194–208.

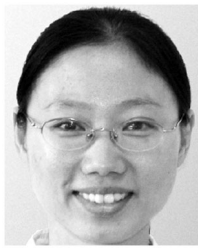
[37] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proc. KDD*, New York, NY, USA, 2000, pp. 169–178.

[38] L. Gravano *et al.*, "Using q-grams in a DBMS for approximate string processing," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 28–34, Dec. 2001.

[39] C. J. van Rijsbergen, *Information Retrieval*. Boston, MA, USA: Butterworth, 1979.

Shaoxu Song is an Assistant Professor in the School of Software, Tsinghua University, Beijing, China. His research interests include data quality and complex event processing.

Lei Chen is an Associate Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. His research interests include crowdsourcing, uncertain databases, data quality, and graph databases.



Hong Cheng is an Assistant Professor in the Department of Systems Engineering and Engineering Management, the Chinese University of Hong Kong, Hong Kong, China. Her primary research interests include data mining, machine learning, and database systems.