Control and Cybernetics

vol. **38** (2009) No. 2

Properties and pre-processing strategies to enhance the discovery of functional dependency with degree of satisfaction*†

by

Qiang Wei, Guoqing Chen and Xiaocang Zhou

Department of Management Science and Engineering,
School of Economics and Management, Tsinghua University
Beijing 100084, China
e-mail: weiq@sem.tsinghua.edu.cn

Abstract: Functional dependency with degree of satisfaction (FD_d) is an extended notion in data modeling, and reflects a type of integrity constraints and business rules on attributes, mainly for massive databases, in which incomplete data such as noise, null and imprecision may exist. While existing approaches are considered effective in general, attempts for further improvement in efficiency are deemed meaningful and desirable as far as knowledge discovery is concerned. This paper focuses on discovering (FD_d)s as a form of useful semantic knowledge, aiming at providing an enhancement to the FD_d mining process in a more efficient manner. In doing so, properties of FD_d are in-depth investigated along with a measure for degree of distinctness. Subsequently, a number of optimization strategies are developed for pre-processing, which are then incorporated into the mining process, giving rise to an enhanced approach for mining functional dependency with degree of satisfaction, namely e-MFDD. Finally, data experiments revealed that e-MFDD significantly outperformed the original approach without pre-processing.

Keywords: functional dependency, incomplete data, degree of distinctness, data mining.

1. Introduction

Nowadays, most of the commercial data repositories for business, engineering and scientific applications including web-based ones are relational databases (RDB). While research on RDB has been intense and long-lasting, on normalization, integrity constraint, Entity-Relationship (ER) conceptual modeling, relational algebra, SQL language, query optimization, etc. (Codd, 1970; Ullman,

*Partly supported by the National Natural Science Foundation of China (70890083/70621061), and Tsinghua Research Center for Contemporary Management.

†Submitted: December 2007; Accepted: October 2008.

1988; Ullman and Widom, 1997), recent years have witnessed a shift in focal point towards large-scale data environments where data generation, storage and use are pervasive, and the volume of information is extraordinarily huge. The implication of the shift is then regarded profound in two respects: one is a need to revisit related theories and applications that may only be valid or sensible with a limited amount of data; the other is to explore new issues that are relevant due to enormous capacities of the IT (information technology)-enabled applications.

Functional dependency (FD) is one of the key notions in the field of data modeling. It is a type of integrity constraints among data attributes on which RDB design theories are based, while reflecting a form of semantic knowledge as business rules of the real world concerned. Generally speaking, an FD indicates a correspondence between two sets of attributes in terms of dependence of one on the other. Concretely, for two collections A and B of attributes, a functional dependency (FD) $A \rightarrow B$ means that A values uniquely determine B values. An example of $A \rightarrow B$ is (Customer#, Product#)→Quantity, meaning that the value of Quantity can be uniquely determined by a given value of Customer# and a given value of Product#. Formally, let A and B be subsets of the attribute set $U = \{I_1, I_2, \ldots, I_m\}$, i.e., $A, B \subseteq U$, $\mathfrak{R}(U)$ be an m-ary relational scheme on domains $D_1, D_2, \ldots, D_m$ with $\mathrm{Dom}(I_i) = D_i$, and R be a relation of scheme $\mathfrak{R}(U)$, $R \subseteq D_1 \times D_2 \times \ldots \times D_m$. A functionally determines B (or B is functionally dependent on A), denoted as $A \rightarrow B$, if and only if

$$\forall\, t,\ t' \in R,\ \text{if}\ t(A) = t'(A)\ \text{then}\ t(B) = t'(B),$$

where t and t' are tuples of R, and $t(A)$, $t'(A)$, $t(B)$ and $t'(B)$ are values of t and t' for A and B respectively. It is important to note that functional dependency possesses several desirable properties, including the so-called Armstrong axioms that constitute a FD inference system (Ullman, 1988):

A1: If $B \subseteq A$, then $A \rightarrow B$;
A2: If $A \rightarrow B$, then $AC \rightarrow BC$;
A3: If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$.

The soundness and completeness of the system (axioms A1, A2, and A3) guarantees, given a set F of functional dependencies (FDs) with respect to $\mathfrak{R}(U)$, the equivalence of two notions, namely F^+ (the set of all FDs logically implied from F in $\mathfrak{R}(U)$) and F^A (the set of all FDs inferred from F with the axioms), (Ullman, 1988; Ullman and Widom, 1997). Here, by FDs being logically implied from F we mean those FDs that hold in all relations of schema $\mathfrak{R}(U)$. This enables RDB designers to normalize the relational schemes and obtain related decomposition algorithms in light of dealing with data redundancy and update anomalies.

In many cases, either due to the improper design of the initial RDB scheme, or due to the insufficient enforcement of FDs while populating/adding data over time, or due to databases re-structuring resulting from various purposes

(e.g., business merges, convenient data handling, intermediate outcome storage, data warehousing etc.), there may exist unknown FDs that are hidden in data and useful in scheme evolution and semantic representation. Some of the FDs could also be used as integrity constraints or business rules for RDB and related information systems to enforce (e.g., via a RDBMS - relational database management system). Thus, as the volume of data gets more and more massive, finding FDs became a subject of research focus, especially in the data mining context (Andersson, 1994; Baudinet, Chomicki and Wolper, 1999; Bell and Brockhausen, 1995; Castellanos and Saltor, 1993; Fayyad, Piatetsky-Shapiro and Smyth, 1996; Flach and Savnik, 1999; Savnik and Flach, 2002; Wijsen, Ng and Calders, 1999; Wyss, Giannella and Robertson, 2001). Moreover, some other attempts centered on extended forms of FD, such as functional dependencies with null values (Liao, Wang and Liu, 1999), partial determination (Kramer and Pfahringer, 1996), approximate functional dependencies (Huhtala et al., 1998, 1999; King and Legendre, 2003; Matos and Grasser, 2004), soft functional dependency (Ilyas et al., 2004), fuzzy functional dependencies (FFDs) (Bhuniya and Niyogi, 1993; Bosc, Dubois and Prade, 1999; Chen, Kerre and Vandenbulcke, 1994, 1996; Chen, Vandenbulcke and Kerre, 1991; Chen, 1998; Cubero et al., 1995, 1999; Kiss, 1991; Kruse, Nanck and Borgelt, 1999; Liu, 1993; Maimon, Kandel and Last, 2001; Mitra, Pal and Mitra, 2002; Mouaddib, 1995; Raju and Majumdar, 1988; Saxena and Tyagu, 1995; Shenoi, Melton and Tan, 1992; Wang, Shen and Hong, 2002; Yang and Singhal, 1999), functional dependency with degree of satisfaction (FD_d) (Wei and Chen, 2003a,b, 2004, 2006; Wei, Chen and Kerre, 2002), etc.

In the environment of massive databases, where the issues addressed in this paper are relevant, incomplete, noisy or distorted data often appear, including conflicts, nulls, and errors that may result from, for instance, inaccurate data entry, transformation or updates. Apparently, these could hardly be tolerated by traditional FDs that by definition strictly express the semantics that "Equal A values determine equal B values (for all tuples)". As a result, any tiny noise would probably lead to negation of FD existence. On the other hand, even without noisy data, sometimes a partial truth of an FD may still make sense. For instance, "an FD almost holds in a database" or "Equal A values determine equal B values for most of the tuples" expresses a sort of partial knowledge, meaning that the FD satisfies the RDB of concern to a certain extent. This gives rise to a notion of functional dependency with degree of satisfaction (FD_d) introduced by Wei, Chen and Kerre (2002).

Related work could be found in Huhtala et al. (1998, 1999), Ilyas et al. (2004), King and Legendre (2003), Matos and Grasser (2004), where some notions of approximate dependency were introduced, including some measures for an error of a dependency and bounds for discovering dependencies with errors. Huhtala et al. (1998, 1999) defined the proximity of a dependency in terms of an error based on the minimum number of rows that need to be removed from the relation. Similar efforts have been made to find approximate dependencies

using an SQL-based algorithm (Matos and Grasser, 2004) and partition-based level-wise algorithm (King and Legendre, 2003). Ilyas et al. (2004) proposed the so-called soft functional dependency, which represents the dependency of one attribute (column) on the other (column), based on statistical correlations with sampling. They also derived several properties to improve the discovery efficiency for their notions of dependencies. However, these notions of dependencies are different from FD_d in form and semantics. Moreover, the Armstrong axioms were not extended in their work and therefore were not used to infer dependencies for optimizing the discovery process.

Overall, the usefulness of discovering functional dependencies (that could have various forms, though) is threefold. First, newly discovered functional dependencies are of interest for modeling purposes. These discovered functional dependencies could be used to refine database schemas/views, where necessary (Hick and Hainaut, 2006; Patig, 2006). For example, some relationships between data attributes (e.g., in long-standing large databases) might not be of FD nature at the stage of database design, which were therefore not enforced thereafter by the system over years. However, some of these relationships satisfied the FD condition starting from a certain point of time due to evolving characteristics of the businesses. Apparently, these relationships are not traditional FDs, in consideration of the whole data over years, but could be represented by (FD_d)s. Hence, after being confirmed by business analysts in the light of FD_d, this change of business semantics may need to be reflected by enforcing the relationships as FDs. Furthermore, in many cases, data warehouses/marts need to be constructed resulting from relevant data attributes of different sources, where the discovered functional dependencies reflecting semantic relationships of these attributes could be useful. Second, these newly discovered functional dependencies are by themselves certain types of associative knowledge, which could be used as business rules for decision making. Third, it is deemed desirable that newly discovered functional dependencies could be used as semantic constraints in query formulation and optimization, especially in the context of high volume information retrieval and search (Hick and Hainaut, 2006). Thus, discovering (FD_d)s as a specific form of functional dependencies is meaningful in the light of noise tolerance and partial knowledge representation.

The rest of this paper is organized as follows. Section 2 will introduce the related notions about functional dependency with degree of satisfaction. Section 3 will discuss Property 4, presented in Section 2, in more detail and introduce the degree of distinctness to measure the "distinctness" of attributes. Then, some related properties will be explored with three optimization strategies being derived so as to improve the efficiency of the FD_d mining process. Section 4 will present an enhanced algorithm that incorporates the three strategies as pre-processing, along with some analysis and discussion on the contribution of the pre-processing strategies in the light of computational complexity. Some experimental results will be discussed in Section 5 to illustrate the effectiveness of the strategies and efficiency of the enhanced algorithm.

2. Related work

Let $\mathfrak{R}(U)$ be a relation scheme, $A, B \subseteq U$, and R be a relation of $\mathfrak{R}(U)$ with n tuples, then an FD_d $(A \to B)_\alpha$, is defined as the degree (denoted as $d_R(A \to B)$, or simply $d(A \to B)$) α at which R satisfies $A \to B$:

$$d(A \to B) = d_R(A \to B) = \frac{\sum\limits_{\substack{\forall\, t_i, t_j \in R \\ t_i \neq t_j}} d_{(t_i,t_j)}(A \to B)}{NTP},$$

where $0 \leq \alpha \leq 1$, NTP represents the number of distinct pairs of tuples in R (which equals $n(n-1)/2$, where n is the number of tuples), and $d_{(t_i,t_j)}(A \to B)$ is the degree that B is functionally dependent on A for the distinct pair of tuples (t_i, t_j) in that if $t_i(A) = t_j(A)$ and $t_i(B) \neq t_j(B)$, then $d_{(t_i,t_j)}(A \to B) = 0$; otherwise 1. In other words, an FD_d $(A \to B)_\alpha$ reflects the semantics that equal B values depend on equal A values at a certain degree (α). Given a minimal satisfaction threshold θ, $0 \leq \theta \leq 1$, if $d(A \to B) \geq \theta$, then $A \to B$ is called a satisfied functional dependency. In a manner analogous to association rule mining, where minimal thresholds are used for degrees of support and confidence in assessing the levels of strength and significance, θ is a minimal threshold for assessing (FD_d)s in terms of the level of dependency tolerance. These thresholds are usually context-related and often pre-specified by business analysts and domain experts (see Agrawal, Imielinski and Swarmi, 1993; Agrawal et al., 1996; and Agrawal and Shafer, 1996).

It is worth mentioning that FD_d is different from the notions of fuzzy functional dependencies. Existing fuzzy functional dependencies are extensions of traditional FDs, focusing on different aspects for generalization (Bosc, Dubois and Prade, 1999; Chen, 1998; Chen, Kerre and Vanderbulcke, 1994; Chen, Vanderbulcke and Kerre, 1991; Cubero et al., 1995; Kiss, 1991; Liao, Wang and Liu, 1999; Liu, 1993; Maimon, Kandel and Last, 2001; Mouaddib, 1995; Raju and Majumdar, 1988; Saxena and Tyagu, 1995; Shenoi, Melton and Tan, 1992; Yang and Singhal, 1999). For instance, 1) fuzziness in attribute values and tuple belongingness could be introduced into RDB models in terms of different fuzzy data representation; 2) if-then truth values in traditional FDs could be extended with fuzzy implication operators (FIOs); 3) equality (=) for attribute values could be extended with similarity/proximity/closeness measures; 4) the degree of dependency between attributes could be assessed in different ways using composition/aggregation methods, giving rise to various definitions of fuzzy functional dependencies. Generally speaking, fuzzy functional dependencies (e.g., A to B) represent semantic relationships such as "similar attribute values of A (approximately) determine similar attribute values of B". As an example, a fuzzy functional dependency $A \to_\phi B$ is defined as:

$$\min_{t,t' \in R} \{I(t(A) \approx t'(A),\, t(B) \approx t'(B))\} \geq \phi,$$

where $A, B \subseteq U$, I is a fuzzy implication operator (FIO) (e.g., Gödel operator), $\approx$ is a closeness measure (reflexive and symmetric), $\phi \in [0, 1]$ (Chen, Kerre and Vanderbulcke, 1996).

Unlike fuzzy functional dependencies, a FD_d $(A \rightarrow B)_\alpha$ represents the semantics that "Equal values of attribute A determine equal values of attribute B for a fraction (α) of tuple-pairs". More importantly, while fuzzy functional dependencies are of fuzzy nature, where membership functions pertain, FD_d is not based on fuzziness-related membership functions but a frequency-related measure. In other words, a FD_d reflects the extent to which a FD holds in a traditional non-fuzzy sense.

Moreover, three important properties of FD_d are proven to hold as a form of extended Armstrong axioms (Wei and Chen, 2004):

A1': If $B \subseteq A$, then $d(A \rightarrow B) = 1$, (PROPERTY 1)

A2': If $d(A \rightarrow B) \geq \alpha$, then $d(AC \rightarrow BC) \geq \alpha$, (PROPERTY 2)

A3': If $d(A \rightarrow B) \geq \alpha$ and $d(B \rightarrow C) \geq \beta$, then $d(A \rightarrow C) \geq \gamma$, where $\alpha + \beta - 1 \leq \gamma \leq 1$. (PROPERTY 3)

These then constitute an FD_d inference system, based on which the notion of FD_d minimal set is presented (Wei and Chen, 2003a,b, 2004, 2006; Wei, Chen and Kerre, 2002). Furthermore, an effective algorithm for mining (FD_d)s (namely MFDD) has been provided by incorporating the properties into the mining process in order to directly infer (FD_d)s and reduce the computational complexity due to scanning of the entire database (Wei and Chen, 2004, 2006).

For illustrative purposes, let us consider an example (Example 1) as follows.

EXAMPLE 1 Table 1 is a Student database, which contains the data about student ID numbers, the departments that students belong to, and the locations of the departments. Note that tuple 1 has a noisy value (denoted as #) in Location. "#" can be erroneous data, such as "bvilding 2", distorted string "*#&@" or null value.

Table 1. Student Database

	ID	**Department**	**Location**
1	001	CS	#
2	002	IS	Building 2
3	003	CS	Building 2
4	004	CS	Building 2
5	005	CS	Building 2

To find traditional functional dependencies, all 6 candidate FDs (i.e., ID→ Department, ID→Location, Department→ID, Department→Location, Location →ID, and Location→Department) need to be examined by scanning the database. By one scan, we mean scanning the relevant attribute of each tuple once.

In other words, one scan is to read every tuple only once, although whether the read is for all its attributes or merely for its relevant attributes depends on the implementation. Thus, after scanning the database 6 times (once for each FD candidate), it could be found that ID→Department and ID→Location hold. Note that Department→Location could not be discovered due to the noisy data value in location, which otherwise ought to hold semantically.

Consider functional dependencies with degrees of satisfaction. Given $\theta = 0.6$, the set of (FD_d)s with respect to this θ value, i.e., $\{(\text{ID} \rightarrow \text{Department})_{1.0}$, $(\text{ID} \rightarrow \text{Location})_{1.0}$, $(\text{Location} \rightarrow \text{Department})_{0.7}$, $(\text{Department} \rightarrow \text{Location})_{0.7}\}$, can be generated using algorithm MFDD, in which Properties 1, 2 and 3 have been integrated to optimize the mining process.

For example, when $(\text{ID}\rightarrow\text{Department})_{1.0}$ and $(\text{Department}\rightarrow\text{Location})_{0.7}$ were discovered, according to Property 3, it could be inferred directly that $d(\text{ID} \rightarrow \text{Location}) \geq \text{d}(\text{ID} \rightarrow \text{Department}) + \text{d}(\text{Department} \rightarrow \text{Location}) - 1 = 1 + 0.7 - 1 = 0.7 \geq \theta$. In other words, ID→Location could be inferred as satisfied without scanning database. Clearly, it is desirable to infer as many (FD_d)s as possible without scanning database, especially with huge m (the number of attributes) and n (the number of tuples). ■

As far as computational efficiency in data mining is concerned, it is usually deemed meaningful and desirable to explore the properties of the relevant notions and develop corresponding algorithmic optimization strategies (such as pruning and inferring) so as to reduce scanning of the databases and respective computations.

In addition to the above three properties, the following property (Property 4) has been proved in Wei and Chen (2004):

PROPERTY 4 *If* $d(A \rightarrow B) = \alpha$, *then* $d(B \rightarrow C) \geq 1 - \alpha$.

In Property 4, $d(A \rightarrow B) = \alpha$ means that the ratio of the cases with $t_i(A) = t_j(A)$ and $t_i(B) \neq t_j(B)$ over total number of distinct pairs of tuples (i.e., NTP) equals $1 - \alpha$. That is to say, the ratio of the cases with $t_i(B) \neq t_j(B)$ is at least $1 - \alpha$. The ratio of the cases with $t_i(B) \neq t_j(B)$ and ($t_i(C) \neq t_j(C)$ or $t_i(C) = t_j(C)$) is at least $1 - \alpha$. This implies $d(B \rightarrow C) \geq 1 - \alpha$ (Definition 1). This property will play an important role in further improvement of computational efficiency of FD_d discovery, which will be discussed in detail in the following sections of the paper.

For instance, according to the definition of traditional FD, if $A \rightarrow B$, e.g., $d(A \rightarrow B) = 1$, it says little about $B \rightarrow C$ (e.g., $d(B \rightarrow C) \geq 0$). On the other hand, in the context of FD_d, if $d(A \rightarrow B) = \alpha < 1$, then $B \rightarrow C$ could be inferred with $d(B \rightarrow C) \geq 1 - \alpha$, which says something and is regarded meaningful. It is also worthwhile to indicate that, with this property, the transitive inference (Property 3) will all have non-negative degrees, that is $\gamma \geq 0$ (i.e., $\gamma \geq \alpha+\beta-1 \geq \alpha + (1 - \alpha) - 1 = 0$).

The focus of this paper is placed on further enhancement of the discovery efficiency in finding (FD_d)s. The main idea for the enhancement is, based on Property 4, to investigate desirable properties along with a measure of degree of distinctness for attributes, and to develop corresponding pre-processing strategies to be integrated with MFDD, resulting in an enhanced algorithm named e-MFDD. The analysis and data experiments will show that e-MFDD effectively outperforms the original MFDD.

3. Degree of distinctness and optimization strategies

As mentioned previously, while the above-mentioned Properties 1, 2 and 3 are used for FD_d inference in original MFDD (Wei and Chen, 2004), Property 4 may further play an important role in the efficiency improvement. Note that in the discussions of this section and thereafter, strategies, which are mainly resulting from properties, are meant as the conditions to be used in algorithmic optimization.

First, Property 4 could be used directly as a strategy in discovering (FD_d)s. Given a threshold θ, A, $B \subseteq U$, we have:

STRATEGY 0 *Given the threshold θ, if $d(A \to B) \leq 1 - \theta$, then $d(B \to Y) \geq \theta$, $\forall Y \in U$.*

Proof. It follows directly from the Property 4. ∎

According to Strategy 0, in the discovery process, if it is discovered that $d(A \to B) \leq 1 - \theta$, then any $B \to Y$ is satisfied (i.e., $d(B \to Y) \geq \theta$), where $Y \in U$. This is quite useful, since it could be used to save many scanning operations for computing the degrees of satisfaction of all ($B \to Y$)s. Moreover, from the perspective of RDB, B could be regarded as a candidate key, since only candidate keys can functionally determine all the other attributes. This carries an important piece of knowledge for further RDB modeling.

Also importantly, Property 4 provides some very interesting information that a dissatisfied FD_d could also be utilized to infer other satisfied (FD_d)s. This is novel and does not hold in the traditional FD context. For instance, in the traditional FD context, if an FD $A \to B$ is not satisfied in a database, it means that for equal A values, there exists at least one pair of B values that are different. However, no more information about B could be deduced: any $B \to Y$ cannot be further inferred, satisfied or not. Instead, in the context of FD_d, the degree of satisfaction (dissatisfaction) could be measured, which could be further utilized in the inference process to improve the efficiency of the discovery. To further illustrate the idea, let us look at Example 2.

EXAMPLE 2 Consider the Student database as shown in Table 1. If Strategy 0 could be integrated in the algorithm MFDD, then since it could be discovered by scanning database that $d(\text{Department} \to \text{ID}) = 0.4 \leq \theta = 0.6$, both

ID→Department and ID→Location are definitely satisfied accordingly. Totally, the computational complexity includes four database scanning operations and two inference operations, which is more efficient than that of MFDD (five times scanning and once inference).

3.1. Degree of distinctness

Intuitively, Property 4 means that if $d(A \to B)$ is sufficiently small ($\leq 1 - \theta$), then the values of B are quite different from each other, which reflects an important characteristic of a candidate key in relational databases. In fact, the level at which the values of any attribute set A are different from each other represents a sort of "distinctness" of A. To measure it, a degree of distinctness is defined as follows (Definition 1).

DEFINITION 1 *Let $\mathfrak{R}(U)$ be a relation scheme, $A \subseteq U$, and R be a relation of $\mathfrak{R}(U)$ with n tuples, then the degree of distinctness in A, denoted by $d_R(A)$ (or $d(A)$ for simplicity), is defined as:*

$$d_R(A) = d(A) = \frac{\sum\limits_{\substack{\forall\, t_i,\, t_j \,\in\, R \\ t_i \neq t_j}} d_{(t_i,t_j)}(A)}{NTP},$$

where NTP represents the number of distinct pairs of tuples in R (which equals $n(n-1)/2$), and $d(t_i, t_j)(A) = 1$, if $t_i(A) \neq t_j(A)$, otherwise 0.

In other words, this measure reflects the degree of pair-wise difference of an attribute (set). Let us use an example to illustrate the idea. Suppose there are n tuples in a database with k distinct values evenly distributed. That is, we can form k distinct groups, each with identical values that are different from the values in another group. Then, we obtain $d(A) = (n/(n-1)) * ((k-1)/k))$. This means that if all the values are identical (i.e., $k = 1$) then $d(A) = 0$; if all the values are mutually distinct (i.e., $k = n$), then $d(A) = 1$. When n is very large, which is common nowadays, we have approximately $d(A) = (k-1)/k$. In this case, for instance, if $k = 2$, 4, 6, 8, 10, then approximately $d(A) = 0.50$, 0.75, 0.83, 0.88, 0.90, respectively. For $k = 2$, this is a case with two distinct groups, each of which accounts for 50% of tuples, which is different from a case with $k = 1$. That means the ratio of 50% over 100%. Likewise, for $k = 4$, we have the ratio of 75% over 100%, and so on. Moreover, $d(A)$ is not only dependent on the number of distinct groups (i.e., k), but also on how the values are distributed among groups. That is to say that $d(A)$ with evenly distributed values is different from $d(A)$ with unevenly distributed values. For example, for $k = 2$, let us have two groups with sizes i and $n - i$, respectively. Then $d(A)$ can be depicted with respect to i/n (in percent) by a curve as shown in Fig. 1.

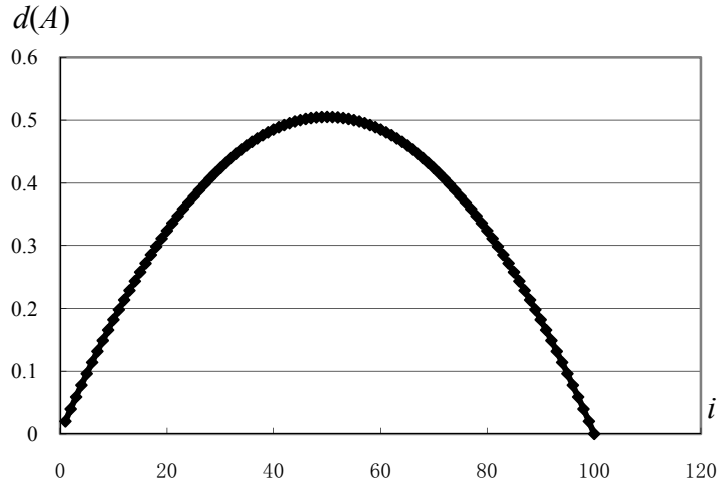

Figure 1. $d(A)$ curve

Fig. 1 indicates that the curve of $d(A)$ is symmetric with its maximal value being at 50%, meaning that the sizes of the two groups are equal (i.e., evenly distributed).

Essentially, the degree of satisfaction of a FD_d (i.e., $d(A \to B)$) is highly related to $d(A)$ and $d(B)$. Generally, the more distinct is A (i.e., $d(A)$ is higher), the higher is the degree of satisfaction of $A \to B$ (i.e., $d(A \to B)$). The less distinct is B, the higher is the degree of satisfaction of $A \to B$ (i.e., $d(A \to B)$). Furthermore, for the Student database example, attribute ID is completely distinct, i.e., the ID values of any two tuples are different. Then, it is a candidate key. In order to take advantage of Property 4, we do not need to calculate any ID$\to Y$, where $Y \subseteq U$. We only need to scan the database and calculate d(ID) before mining (FD_d)s, and if ID is distinct enough, then it could be inferred that ID$\to$Department and ID$\to$Location are satisfied.

Note that if A values of all tuples are identical, then $d(A) = 0$, meaning that the attribute set A is not distinct in value at all. If the A value of each tuple is unique (mutually distinct), then $d(A) = 1$, meaning that attribute set A is totally distinct in value, which also indicates that A is a candidate key. Further, if there exists an attribute set $X \subseteq U$, which is totally distinct, then we have a number of relationships between the degree of distinctness and the degree of satisfaction as follows.

PROPERTY 5 $\forall A,\ X \subseteq U$, *if* $d(X) = 1$ *then* $d(A) = d(A \to X)$.

Proof. Without loss of generality, all the distinct pairs of tuples (i.e., $\{(t_i, t_j) | t_i, t_j$

$\in R, t_i \neq t_j\}$) could be classified into two groups, namely, G_1 and G_2:

$$G_1 = \{(t_i, t_j) | t_i, t_j \in R,\ t_i \neq t_j,\ t_i(A) = t_j(A)\};$$
$$G_2 = \{(t_i, t_j) | t_i, t_j \in R,\ t_i \neq t_j,\ t_i(A) \neq t_j(A)\}.$$

Since $d(X) = 1$, then $\forall t_i, t_j \in R$, $t_i(X) \neq t_j(X)$. Subsequently, by definition, $d_{(t_i,t_j)}(A) = 1$ if $t_i(A) \neq t_j(A)$, otherwise 0. Furthermore, by definition, $d_{(t_i,t_j)}(A \to X) = 1$ if $t_i(A) \neq t_j(A)$ and $t_i(X) \neq t_j(X)$, or $d_{(t_i,t_j)}(A \to X) = 0$ if $t_i(A) = t_j(A)$ and $t_i(X) \neq t_j(X)$. This could be summarized in the following table with $|G_1| = N_1$ and $|G_2| = N_2$.

Group of distinct pairs of tuples	**Number of distinct pairs of tuples**	$t_i(A) \neq t_j(A)$	$t_i(X) \neq t_j(X)$	$d_{(t_i,t_j)}(A)$	$d_{(t_i,t_j)}(A \to X)$
G_1	N_1	No	Yes	0	0
G_2	N_2	Yes	Yes	1	1

Hence, according to the definitions of $d(A)$ and $d(A \to X)$, we have $d(A \to X) = N_2/(N_1 + N_2) = d(A)$. ∎

Moreover, one can easily get, $\forall Y \in U$, $d(X \to Y) = 1$ if $d(X) = 1$.

Based on Definition 1 and the above-mentioned properties, some important derivatives could be further obtained. Let R be a relation on $\mathfrak{R}(U)$ and $A, B \subseteq U$, then we have the following properties (Properties 6 and 7).

PROPERTY 6 $d(AB) \geq d(A)$.

Proof. If $\exists X \subseteq U$, $d(X) = 1$, then, according to Property 5, $d(AB) = d(AB \to X)$ and $d(A) = d(A \to X)$. If $X \not\subset U$, we could easily construct $U' = U \cup X$, such that $d(X) = 1$. In this case, AB, $X \subseteq U'$, then again, according to Property 5, $d(AB) = d(AB \to X)$ and $d(A) = d(A \to X)$.

According to Properties 2 and 3,

$$d(AB) = d(AB \to X) \geq d(AB \to A) + d(A \to X) - 1 \geq 1$$
$$+ d(A \to X) - 1 = d(A \to X) = d(A).$$

So $d(AB) \geq d(A)$. ∎

PROPERTY 7 $d(AB) \leq d(A) + d(B)$.

Proof. Similarly to the proof of Property 6 for dealing with X, $d(AB) = d(AB \to X)$, $d(A) = d(A \to X)$ and $d(B) = d(B \to X)$. According to Properties 1, 2 and 3, we have

$$d(A \to X) \geq d(A \to AB) + d(AB \to X) - 1 = d(A \to B) + d(AB \to X) - 1.$$

According to Property 4, we have $d(A \to B) + d(B \to X) \geq 1$, so $d(A \to B) \geq 1 - d(B \to X)$. Then we have $d(A \to X) \geq 1 - d(B \to X) + d(AB \to X) - 1$. Thus, $d(A \to X) + d(B \to X) \geq d(AB \to X)$. Therefore, $d(A) + d(B) \geq d(AB)$ (Property 5). ∎

3.2. Optimization strategies

Based on the properties discussed in the previous subsection, this subsection mainly concentrates on optimization strategies that will be used in the algorithmic enhancement to be described in greater detail in later sections.

PROPERTY 8 $d(A \to B) \geq d(A)$.

Proof. Similarly to the proof of Property 6 for dealing with X, $d(A) = d(A \to X)$. Further, $d(A \to B) \geq d(A \to X) + d(X \to B) - 1$ (Property 3). Since $d(X \to B) = 1$, then $d(A \to B) \geq d(A \to X) = d(A)$. Therefore, $d(A \to B) \geq d(A)$. ∎

PROPERTY 9 $d(A \to B) \geq 1 - d(B)$.

Proof. Similarly to the proof of Property 6 for dealing with $X, d(B) = d(B \to X)$. Further, $d(A \to B) + d(B \to X) \geq 1$, according to Property 4. Then $d(A \to B) \geq 1 - d(B \to X) = 1 - d(B)$. That is $d(A \to B) \geq 1 - d(B)$. ∎

PROPERTY 10 $d(A \to B) \leq d(A) + 1 - d(B)$.

Proof. Similarly to the proof of Property 6 for dealing with X, $d(A) = d(A \to X)$ and $d(B) = d(B \to X)$. Further, $d(A \to X) \geq d(A \to B) + d(B \to X) - 1$, according to Property 3. Then $d(A) \geq d(A \to B) + d(B) - 1$. That is, $d(A \to B) \leq d(A) + 1 - d(B)$. ∎

Note that with Properties 8, 9 and 10, if all the attributes could be examined to compute the degrees of distinctness prior to the mining process of MFDD, then some (FD_d)s could be inferred satisfied or dissatisfied without scanning the database. Accordingly, some algorithmic optimization strategies could be developed as follows.

STRATEGY 1 *If $d(A) \geq \theta$, then $d(A \to Y) \geq \theta$, which means that $A \to Y$ is satisfied, $\forall Y \in U$.*

Proof. It can be directly obtained based on Property 8. ∎

Strategy 1 means that, for an FD_d $A \to Y$, if A values are sufficiently distinct (i.e., $d(A) \geq \theta$), then $A \to Y$ is satisfied (i.e., $d(A \to Y) \geq \theta$), whatever Y is. This is because the degree of satisfaction is mainly determined by distinct A values. A could be considered as a candidate key of the relation.

STRATEGY 2 *If $1 - d(A) \geq \theta$, then $d(Y \to A) \geq \theta$, which means that $Y \to A$ is satisfied, $\forall Y \in U$.*

Proof. It can be directly obtained based on Property 9. ∎

Strategy 2 means that, for an FD_d $Y \to A$, if A values are not so distinct (i.e., $1-d(A) \geq \theta$), then $Y \to A$ is satisfied (i.e., $d(Y \to A) \geq \theta$), whatever Y is. This is because the degree of satisfaction is mainly determined by equal A values.

STRATEGY 3 *If $d(B) - d(A) > 1 - \theta$, then $d(A \to B) < \theta$, which means that $A \to B$ is dissatisfied.*

Proof. Since $d(B) - d(A) > 1 - \theta$, then $d(B) - 1 - d(A) > 1 - \theta - 1$, which is $d(A)+1-d(B) < \theta$. Further, according to Property 10, $d(A \to B) \leq d(A) + 1 - d(B) < \theta$, which is $d(A \to B) < \theta$. Therefore, $A \to B$ is dissatisfied. ∎

Strategy 3 means that, for an FD_d $A \to B$, if B values are much more distinct than A values (i.e., $d(B) - d(A) > 1 - \theta$), then $A \to B$ will be dissatisfied (i.e., $d(A \to B) < \theta$).

These three strategies could be used to form a set of pre-processing operations incorporated with MFDD, which enables us to infer as many (FD_d)s as possible prior to the mining process with MFDD, and therefore to considerably improve efficiency. Moreover, it is worthwhile to indicate that Strategy 0 could be replaced by Strategy 1. That is, for strategy 0, if $d(A \to B) \leq 1 - \theta$, then $d(A \to B) \geq 1 - d(B)$ (according to Property 9). Further, we have $1 - d(B) \leq 1 - \theta$, so $d(B) \geq \theta$, then $d(B \to Y) \geq \theta$ (according to Strategy 1), where $Y \in U$. In other words, if an FD_d can be inferred by Strategy 0, then this FD_d can be inferred by Strategy 1. Therefore, it suffices to use Strategy 1 without further considering Strategy 0.

4. Procedure for pre-processing operations

The algorithmic details of the procedure for pre-processing operations with the above mentioned Strategies 1, 2 and 3 are shown in Table 2.

Consider the computational efficiency of the procedure. First, if $d(A)$ is large ($\geq \theta$) enough, which represents the fact that the attribute set A is quite distinct and could be a candidate key of the database, Strategy 1 will take effect. Then, $A \to Y$, $\forall Y \in U$, will be inferred satisfied. The number of inferred satisfied (FD_d)s will be $|U| - 1 = m - 1$, which is a quite effective inference process. Here, the smaller θ, the more effective Strategy 1 will be.

Second, if $d(A)$ is small ($\leq 1 - \theta$) enough, Strategy 2 will take effect. Then $Y \to A$, $\forall Y \in U$, will be inferred satisfied. The number of inferred satisfied (FD_d)s will be $m-1$, which is also a quite effective inference process. Similarly, the smaller θ is, the more effective Strategy 2 will be.

Third, if $d(B) - d(A) > 1 - \theta$, then $A \to B$ is dissatisfied. At a first glance, compared with Strategies 1 and 2, Strategy 3 is seemingly not so effective, since it could only infer one FD_d. In real applications, however, to guarantee the reliability of discovered (FD_d)s, θ may be set to a quite large value (e.g., significantly greater than 0.5). In this case, it is quite likely that $d(B) - d(A)$ will be bigger than $1 - \theta$. That is, Strategy 3 will also make a significant effect

Table 2. Procedure for pre-processing operations

Procedure for pre-processing operations

```
for A ∈ U { Calculating d(A);                    // Scan and calculate d(A)
for A ∈ U
{    if d(A) ≥ θ
     {    for X ∈ (U − A)
          {    Mark A → X as inferred satisfied;
               d(A → X) = d(A);
          }
     }
}                                                // Strategy 1
for A ∈ U
{    if 1 − d(A) ≥ θ
     {    for X ∈ (U − A)
          {    Mark X → A as inferred satisfied;
               d(X → A) = 1 − d(A);
          }
     }
}                                                // Strategy 2
for A ∈ U
{    for B ∈ U
     {    if d(B) − d(A) ≥ 1 − θ
          {    Mark A → B as inferred dissatisfied;
               d(A → B) = 0;
          }
     }
}                                                // Strategy 3
```

in inference. Here, the larger θ, the more effective Strategy 3 will be. This is shown by data experiments in Section 5.

Importantly, given θ, the whole mining process (pre-processing + MFDD) will be quite efficient when θ is either small or large. This is a novel feature. The time complexity of MFDD without pre-processing is increasing with the increase of θ. This is because, as θ increases, the number of inferred (FD_d)s decreases, leading to more database scans. This situation can be improved by introducing the pre-processing operations prior to MFDD. First, Strategies 1 and 2 will infer more satisfied (FD_d)s when θ is small, which saves the time needed to scan databases for finding the satisfied (FD_d)s. Second, Strategy 3 will infer more dissatisfied (FD_d)s when θ is large, which saves the time needed to scan databases for finding the dissatisfied (FD_d)s. Overall, the time complexity curve along θ for the whole mining process is roughly $\cap$-shaped.

As expressed previously, θ is a minimal threshold for assessing (FD_d)s in terms of the level of dependency tolerance. The setting of θ is usually context-related and often pre-specified by business analysts and domain experts. The above discussion shows that the three strategies would perform differently in inferring (FD_d)s at different levels of θ.

More specifically, the cost of pre-processing is to scan m attributes with n tuples, i.e., $O(m \times n^2)$, usually $m << n$. However, for an FD_d $A \rightarrow B$, the time complexity of calculating its degree of satisfaction is $O(2 \times n^2)$, due to the database scanning operation. Roughly, if $m/2$ (FD_d)s could be inferred (satisfied or dissatisfied) after pre-processing, the efficiency of the whole mining process will be improved, which is demonstrated by real data experiments to be discussed in Section 5.

It is worth indicating that there are similarities and differences with respect to association rule mining. On the one hand, both association rules and functional dependencies, such as (FD_d)s, are associative knowledge, which are of interest for discovery. In the context of massive data, both mining approaches apply optimization strategies to reducing the computational complexity. An important strategy is to use the lattice-type relationship to reduce the search space in rule/dependency generation. On the other hand, these two types of associative knowledge are different in form and semantics. The corresponding optimization strategies are developed depending upon their specific notions. For instance, in association rule mining, apriori-type approaches are commonly used (Agrawal, Imielinski and Swarmi, 1993; Agrawal et al., 1996; Agrawal and Shafer, 1996), while in FD_d discovery, extended Armstrong axioms are used, resulting in different technical treatment and implementation procedures.

5. Data, experimental results and analysis

In order to examine the effectiveness of the pre-processing operations with MFDD, we tested it on some benchmarking data sets available in UCI Machine Learning Repository (Merz and Murphy, 1996), which are widely used in data mining performance evaluation. Five datasets were selected in consideration of reasonable size (e.g., $>$ 100 tuples), null value proportion (no less than 30%) and application variety. Table 3 shows information about the five datasets. The experiments were conducted with θ ranging from 0.50 to 1.00 (which is considered reasonable in usual applications where noisy/null data are present) and in the environment of a Pentium IV 3.0GHz computer, with 1G RAM and Visual Studio 2005 on Windows XP platform.

In order to clearly investigate the performance of the three pre-processing strategies, we ran the experiments with five algorithms, namely, "Original" (MFDD), "S1" (Strategy 1 + MFDD), "S2" (Strategy 2 + MFDD), "S3" (Strategy 3 + MFDD), and "All" (e-MFDD: Strategies 1, 2, 3 + MFDD).

In the light of the previous analysis (Wei and Chen, 2004), computational complexity is mainly attributed to scanning databases. For illustrative pur-

Table 3. Information about the five datasets used in the experiments

No.	Dataset	Number of Attributes	Number of Tuples
1	Led-Display-Creator	7	3200
2	Zoo	18	101
3	Lymph	19	148
4	Machine	10	209
5	Solar	13	323

poses, we focused on the number of those (FD_d)s that could only be determined, whether satisfied or dissatisfied, by scanning databases, called Scanned (FD_d)s. All the other (FD_d)s could be determined, whether satisfied or dissatisfied, by inference without scanning databases. Moreover, the running times of the five algorithms were also compared. Figs. 2-3, 4-5, 6-7, 8-9, and 10-11 present the numbers of scanned (FD_d)s and the running times for the five datasets, respectively.

Notably, the curves of the numbers of scanned (FD_d)s were quite similar, in shape, to the curves of the running times. This observation reflects the fact that the computation mainly depends on database scanning rather than on FD_d inference. Strategy 1 was effective and reduced the running times, especially when θ was small. Strategy 2 was effective and significantly reduced the running times, especially when θ was not large. Strategy 3, as analyzed previously, was effective, too, and significantly reduced running times, especially when θ was large. Furthermore, as an aggregate effect, the process with three strategies (i.e., e-MFDD, denoted as "All" in the figures) was very effective in lowering down the whole computational complexity and outperformed the other four algorithms, with the curves of both the numbers of scanned (FD_d)s and the running times showing to be somewhat $\cap$-shaped.

Moreover, in order to further illustrate the effect of aggregating the three strategies, a real-world database was used for the enhanced algorithm (e-MFDD). The database contains the real business data of The Insurance Company (TIC) Benchmark provided by Dutch Data Mining Company Sentient Machine Research, which is usually used as benchmarking data for evaluating data mining algorithms. The database is from "The Insurance Company 2000," containing 5,822 tuples with 86 attributes (*The Insurance Company, 2000*). For illustrative purposes and the sake of simplicity, we projected the whole dataset on the first 10 attributes ($m = 10$, $n = 5{,}822$). Each tuple represents some of the customer's basic information, e.g., Customer Subtype, Customer Main Type, Number of Houses, Age, Married, etc.

First, pre-processing was carried out with the degrees of distinctness computed as shown in Table 4.

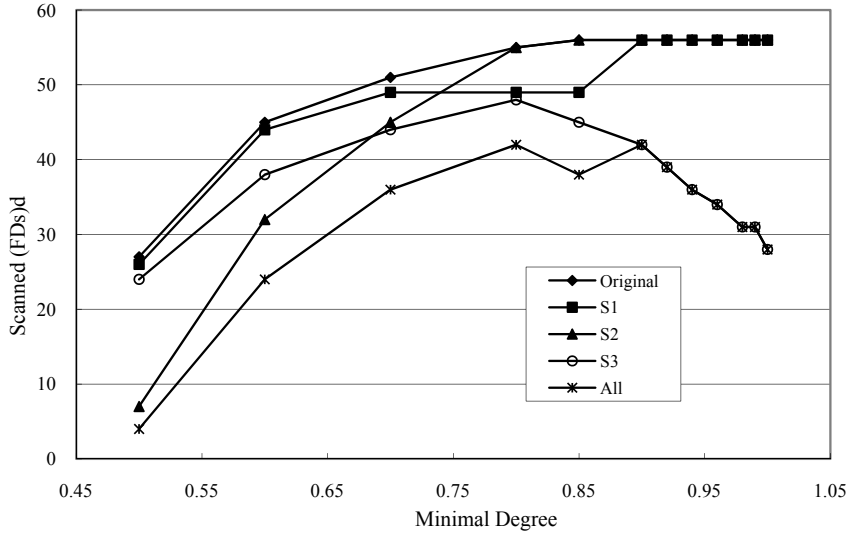

Figure 2. Numbers of Scanned (FD_d)s for the Led-Display Creator dataset

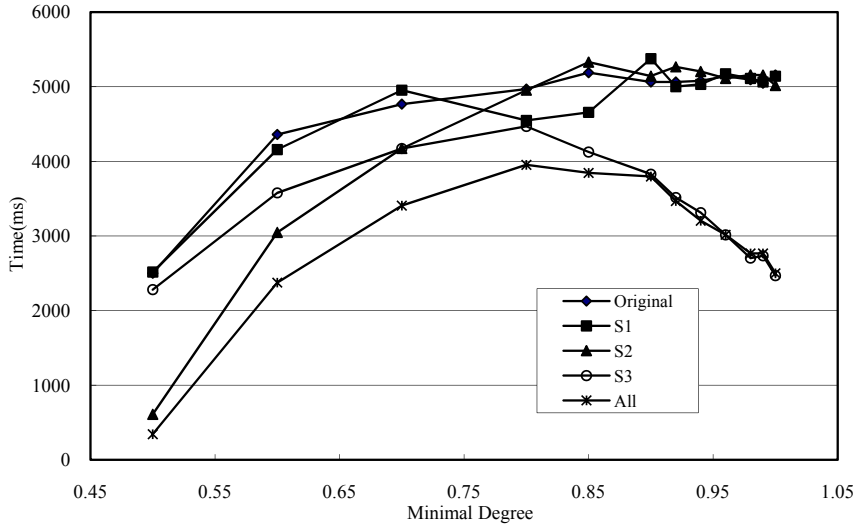

Figure 3. Running times for the Led-Display Creator dataset

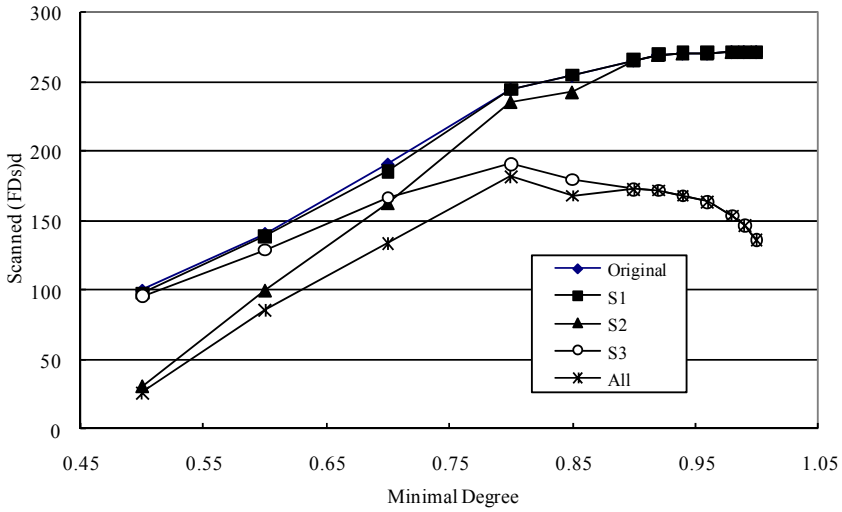

Figure 4. Numbers of Scanned (FD_d)s for the Zoo dataset

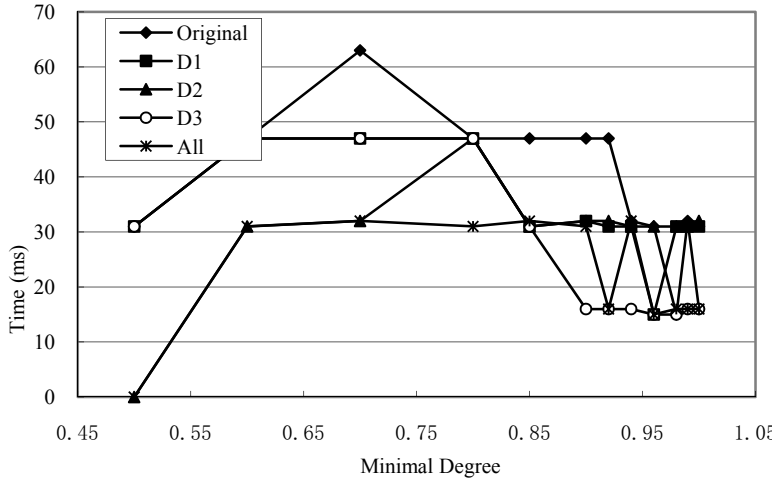

Figure 5. Running times for the Zoo dataset

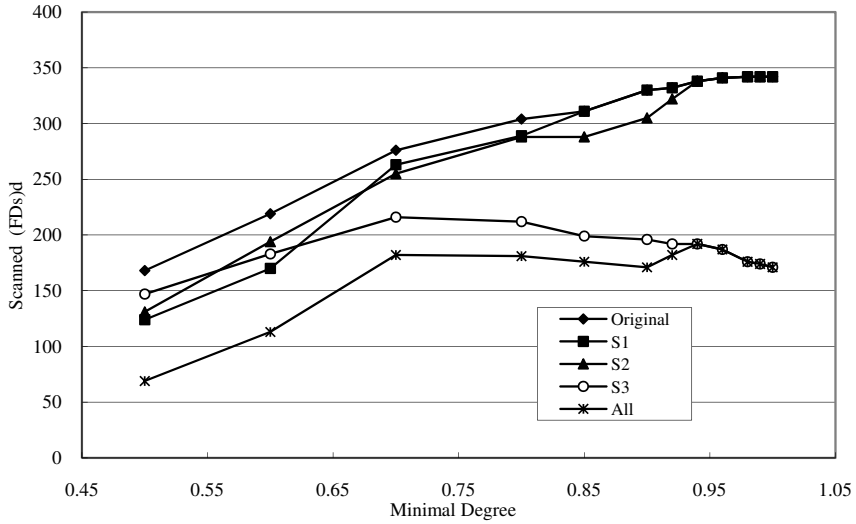

Figure 6. Numbers of Scanned (FD_d)s for the Lymph dataset

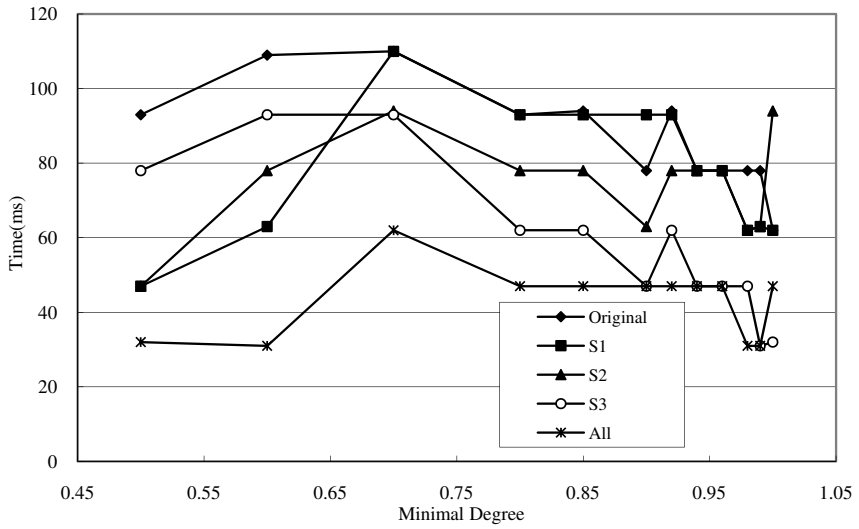

Figure 7. Running times for the Lymph dataset

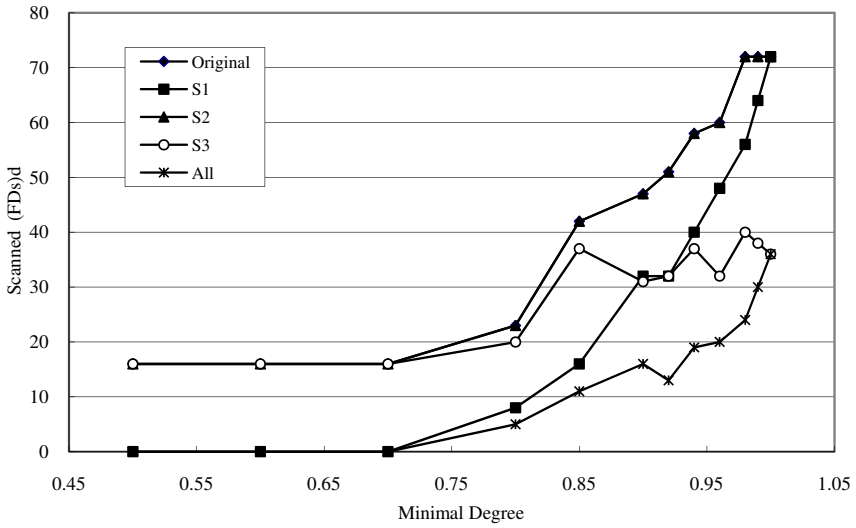

Figure 8. Numbers of Scanned (FD_d)s for the Machine dataset

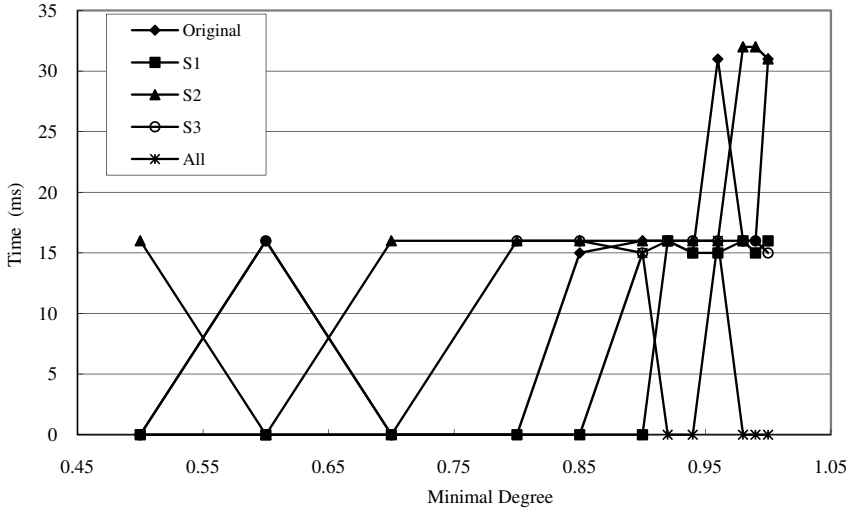

Figure 9. Running times for the Machine dataset

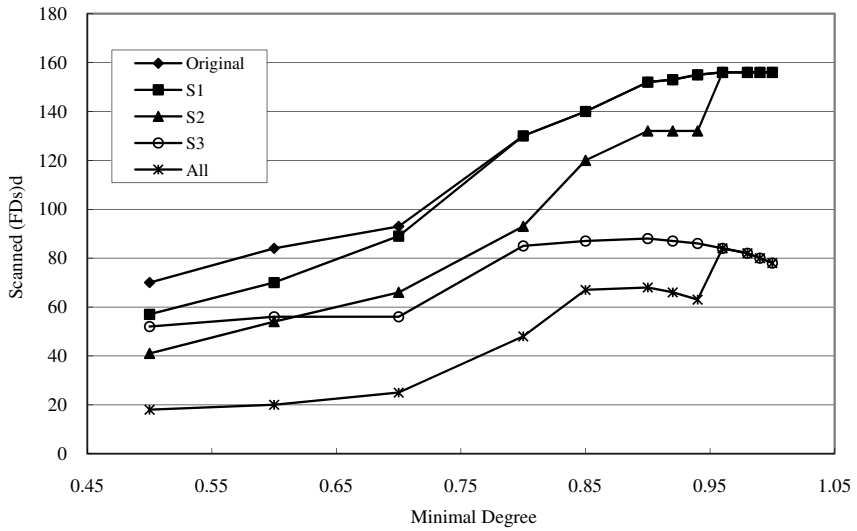

Figure 10. Numbers of Scanned (FD_d)s for the Solar dataset

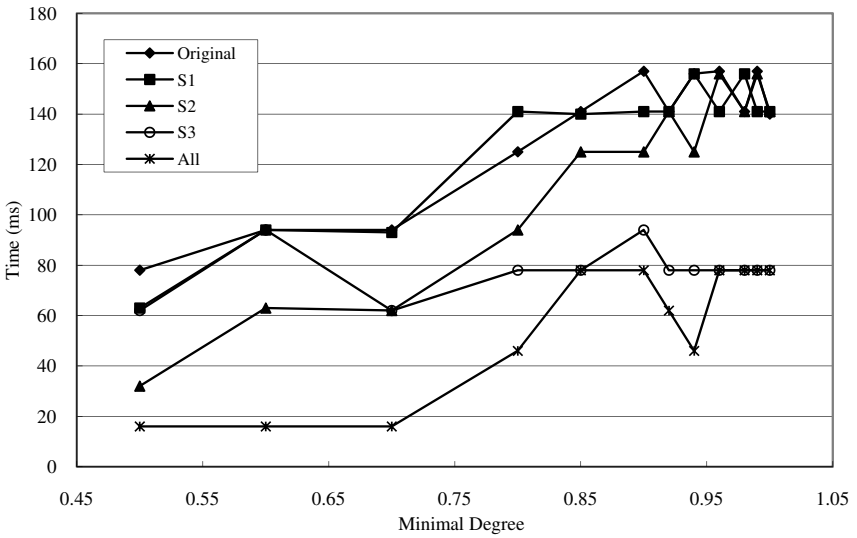

Figure 11. Running times for the Solar dataset

Table 4. Degrees of distinctness ($m = 10$, $n = 1{,}000$)

Attribute #	Attribute Name	Degree of Distinctness
I_1	Customer Subtype	0.950
I_2	Number of Houses	0.175
I_3	Average Size Household	0.643
I_4	Average Age	0.634
I_5	Customer Main Type	0.849
I_6	Roman Catholic	0.595
I_7	Protestant	0.821
I_8	Other Religion	0.699
I_9	No Religion	0.818
I_{10}	Married	0.822

Table 4 indicates that most of the attributes had relatively high degrees of distinctness, especially attribute I_1 with $d(I_1) = 0.950$. In fact, I_1 is "Customer Subtype", which to a large extent could be deemed as a candidate key of the database. It can be seen that Strategy 1 took effect in inferring $(I_1 \rightarrow Y)$, $\forall Y \in U$, as satisfied. Then, attribute I_2 had a relatively low degree, $d(I_2) = 0.175$, where Strategy 2 took effect in inferring $(Y \rightarrow I_2)$, $\forall Y \in U$, as satisfied. Moreover, because of the largely distinct degrees for attribute I_2 and other attributes, it could be expected that Strategy 3 was effective in inferring $(I_2 \rightarrow Y)$, $\forall Y \in U$, as dissatisfied. Figs. 12 and 13 visualize the experimental results.

Apparently, algorithm e-MFDD ("All") significantly improved the efficiency. In addition, the discovery outcomes appeared to be intuitively appealing. For instance, given $\theta = 0.95$, the discovered satisfied (FD_d)s are shown in Table 5.

It is worth mentioning that the traditional notion of FD would only result in Customer Subtype→Customer Maintype (i.e., #8 in Table 5), whereas other attributes (i.e., #1, 6, 7, 9, 10, 11, 12 and 13 in Table 5) could hardly be considered to depend on Customer Subtype, in the traditional FD sense, due to the existence of noisy data in the database, which, however, might be regarded to hold semantically otherwise. On the other hand, the high degree of dependency on Customer Subtype was identified in terms of discovered (FD_d)s. More importantly, (FD_d)s 1, 6, 7, 8, 9, 10, 11, 12 and 13 could be directly inferred with Strategy 1, since d(Customer Subtype) ≥ 0.95. Further, in this example, since $1 - d$(Number of Houses) $= 0.825 < \theta = 0.95$, Strategy 2 did not take any effect, which is reflected in Fig. 11. Finally, many dissatisfied (FD_d)s (they were not included in the set of discovered (FD_d)s) were pruned out according to Strategy 3, which is reflected in Fig. 11. Overall, it can also be seen that e-MFDD with all three strategies ("All") were superior, in the light of computational efficiency, to other algorithms concerned (i.e., "Original", "S1", "S2", and "S3").

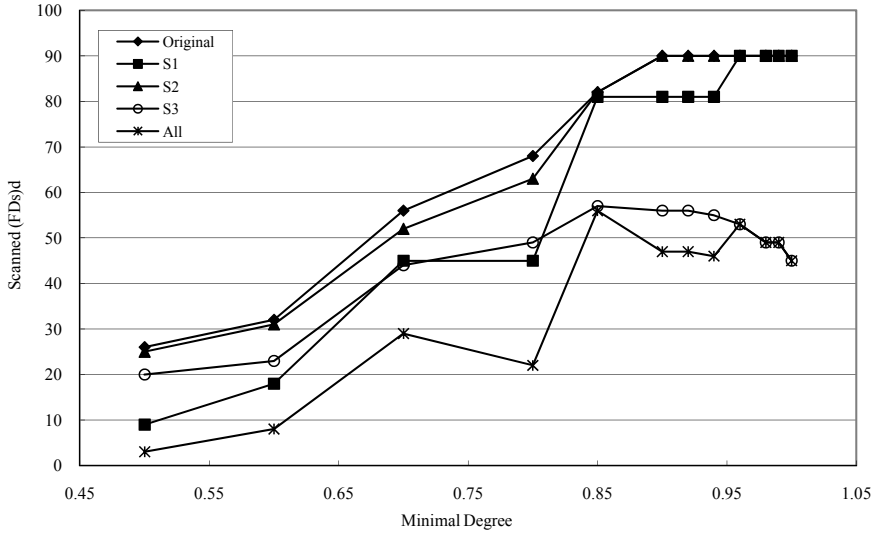

Figure 12. Numbers of Scanned (FD_d)s

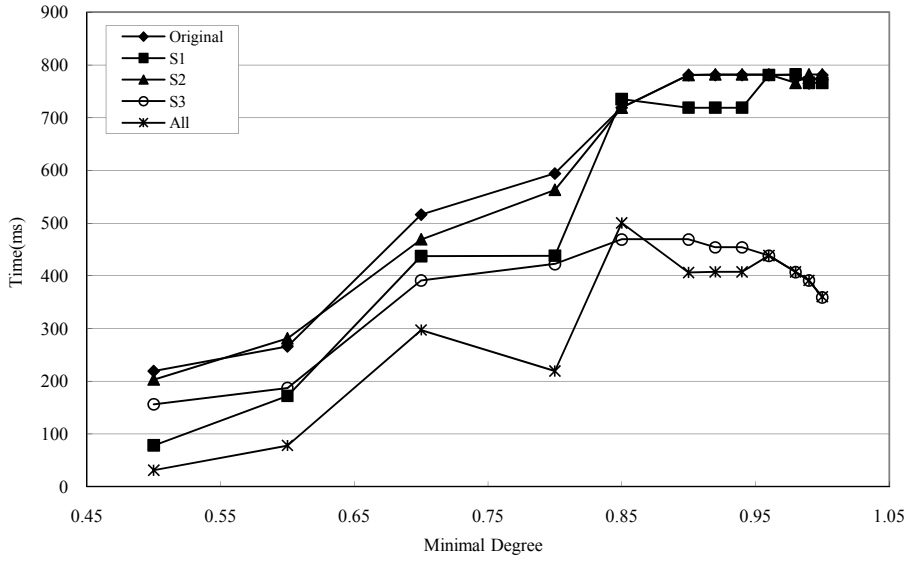

Figure 13. Running times

Table 5. Discovered Satisfied (FD_d)s with $\theta = 0.95$

#	**Discovered Satisfied (FD_d)s**	**Degree of Satisfaction**
1.	Customer Subtype→Number of Houses	0.99
2.	Customer Maintype→Number of Houses	0.98
3.	Protestant→Number of Houses	0.97
4.	No Religion→Number of Houses	0.97
5.	Married→Number of Houses	0.97
6.	Customer Subtype→Average Size Household	0.97
7.	Customer Subtype→Average Age	0.97
8.	Customer Subtype→Customer Maintype	1.00
9.	Customer Subtype→Roman Catholic	0.98
10.	Customer Subtype→Protestant	0.96
11.	Customer Subtype→Other Religion	0.97
12.	Customer Subtype→No Religion	0.96
13.	Customer Subtype→Married	0.96

6. Conclusions

Discovery of functional dependency with degree of satisfaction (FD_d) is considered meaningful and necessary in business modeling and RDB design, especially in the context of massive datasets where incomplete data (such as noisy data) appear. Based upon the concept of FD_d meant to tolerate possible noises in data as well as to reflect valuable partial knowledge of dependency among data attributes, this paper has proposed an efficient enhancement to discovering (FD_d)s. The proposed approach along with the corresponding mining algorithm (namely e-MFDD) has resulted from introducing the measure of degree of distinctness, obtaining a number of related important properties, and developing three algorithmic pre-processing strategies that have then been incorporated into the whole mining process so as to optimize the FD_d discovery in reducing the computational complexity. The theoretical analysis and data experiments have revealed that the properties were desirable, the corresponding strategies were effective, and the enhanced algorithm (e-MFDD) was well advantageous over other ones in improving the efficiency.

The future effort will be in extending the approach to deal with possible closeness relation in data where each distinct pair of tuples may be partially equal. This extended scope of research focus on data and discovery is then generally in the realms of possibility-based database modeling and of fuzzy data mining in that imprecise attribute values and/or close domain elements are assumed, and the respective knowledge discovered (e.g., FD_d) would be semantically richer. Another possible subject for future research is to explore

the possibility of database modeling using discovered (FD_d)s, where relational (de)composition needs to be further investigated (Berzal et al., 2002).

References

Agrawal, R., Imielinski, T. and Swarmi, A. (1993) Mining Association Rules between Sets of Items in Large Databases. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, DC, USA. ACM Press, 207-216.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A.I. (1996) Fast Discovery of Association Rules. In: U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, MA, USA, 1-30.

Agrawal, R. and Shafer, J. (1996) Parallel Mining of Association Rules. *IEEE Transactions on Knowledge and Data Engineering* **8**, 962-969.

Andersson, M. (1994) Extracting an Entity Relationship Schema from a Relational Database through Reverse Engineering. *Lecture Notes in Computer Science* **881**, *Proceedings of the 13th International Conference on the Entity-Relationship Approach*. Springer Verlag, London, 403-419.

Baudinet, M., Chomicki, J. and Wolper, P. (1999) Constraint-generating dependencies. *Journal of Computer and System Science* **59** (1), 94-115.

Bell, S. and Brockhausen, P. (1995) Discovery of Data Dependencies in Relational Databases. University of Dortmund, Computer Science Department, LS-8 Report 14.

Berzal, F., Cubero, J.C., Cuenca, F. and Medina, J.M. (2002) Relational decomposition through partial functional dependencies. *Data & Knowledge Engineering* **43** (2), 207-234.

Bhuniya, B. and Niyogi, P. (1993) Lossless join property in fuzzy relational databases. *Data & Knowledge Engineering* **11** (22), 109-124.

Bosc, P., Dubois, D. and Prade, H. (1999) Fuzzy functional dependencies and redundancy elimination. *Journal of the American Society for Information Science*, **49** (3), Special Issue: *Management of Imprecision and Uncertainty*, Published Online, 217-235.

Castellanos, M. and Saltor, F. (1993) Extraction of Data Dependencies. *European-Japanese Conf. on Information Modelling and Knowledge Bases*, Budapest, Hungary, May 31-June 3. IOS Press, Amsterdam, 401-421.

Chen, G.Q. (1998) *Fuzzy Logic in Data Modeling: Semantics, Constraints and Database Design*. Kluwer Academic Publishers, Boston.

Chen, G.Q., Kerre, E.E. and Vandenbulcke, J. (1994) A computational algorithm for the FFD closure and a complete axiomatization of fuzzy functional dependency (FFD). *International Journal of Intelligent Systems* **9**, 421-439.

Chen, G.Q., Kerre, E.E. and Vandenbulcke, J. (1996) Normalization Based on Fuzzy Functional Dependency in a Fuzzy Relational Data Model. *Information Systems* **21** (3), 299-310.

Chen, G.Q., Vandenbulcke, J. and Kerre, E.E. (1991) A step towards the theory of fuzzy relational database design. In: B. Loewen, M. Roubens, eds., *Proc. of the 4^{th} IFSA World Congress*, Brussels, 44-47.

Codd, E.F. (1970) A Relational Model for Large Shared Data Banks. *Communications of the ACM* **13** (6), 377-387.

Cubero, J.C. et al. (1999) Data Summarization in Relational Databases through Fuzzy Dependencies, *Information Sciences* **121** (3-4), 233-270.

Cubero, J.C., Medina, J.M., Pons, O. and Vila, M.A. (1995) Rules discovery in fuzzy relational databases. In: *Conference of the North American Fuzzy Information Processing Society, NAFIPS'95.* Maryland (USA). IEEE Computer Society Press, 414-419.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996) From Data Mining to Knowledge Discovery: An Overview. In: U, Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, eds., *Advances in Knowledge Discovery and Data Mining*, Cambridge, MA: AAAI Press/The MIT Press, U.S.A., 1-30.

Flach, P.A. and Savnik, I. (1999) Database Dependency Discovery: A Machine Learning Approach. *AI Communications* **12** (3), 139-160.

Hick, J.M. and Hainaut, J.L. (2006) Database Application Evolution: A Transformational Approach. *Data & Knowledge Engineering* **59**, 534-558.

Huhtala, Y., Karkkainen, J., Porkka and P., Toivonen, H. (1999) TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *The Computer Journal* **42** (2), 100-111.

Huhtala, Y., Karkkainen, J., Porkka, P. and Toivonen, H. (1998) Efficient Discovery of Functional and Approximate Dependencies Using Partitions. *Proc. 14th Int. Conf. on Data Engineering*, IEEE Computer Society Press.

Ilyas, I.F., Markl, V., Haas, P., Brown, P. and Aboulnaga, A. (2004) CORDS: automatic discovery of correlations and soft functional dependencies. *International Conference on Management of Data. Proceedings of the 2004 ACM SIGMOD*, Paris, France. ACM Press, 647-658.

King, R.S. and Legendre, J.J. (2003) Discovery of functional and approximate functional dependencies in relational databases. *Journal of Applied Mathematics and Decision Sciences* **7** (1), 49-59.

Kiss, A. (1991) λ-decomposition of fuzzy relational database. *Annales Univ. Sci. Budapest., Sect. Comp.*

Kramer, S. and Pfahringer, B. (1996) Efficient Search for Strong Partial Determinations. *KDD 1996.* AAAI Press, 371-374.

Kruse, R., Nanck, D. and Borgelt, C. (1999) Data Mining with Fuzzy Methods: Status and Perspectives. In: *Proc. 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, CD-ROM.

Liao, S.Y., Wang, H.Q. and Liu, W.Y. (1999) Functional Dependencies with Null Values, Fuzzy Values, and Crisp Values. *IEEE Transactions on Fuzzy Systems* **7** (1), 97-103.

Liu, W. (1993) The fuzzy functional dependency on the basis of the semantic distance. *Fuzzy Sets and Systems* **59** (2), 173-179.

Maimon, O., Kandel, A. and Last, M. (2001) Information-Theoretic Fuzzy Approach to Knowledge-Discovery in Databases. In: R. Roy, T. Furuhashi and P.K. Chawdhry, eds., *Advances in Soft Computing - Engineering Design and Manufacturing*, Springer-Verlag, London, 315-326.

Matos, V. and Grasser, B. (2004) SQL-based discovery of exact and approximate functional dependencies. *ACM SIGCSE Bulletin* **36** (4), 58-63.

Merz, C.J. and Murphy, P. (1996) UCI repository of machine learning databases (http://www.cs.uci.edu/~mlearn/MLRepository.html)

Mitra, S., Pal, S.K. and Mitra, P. (2002) Data Mining in Soft Computing Framework: A Survey. *IEEE Transactions on Neural Networks* **13** (1), 3-14.

Mouaddib, N. (1995) Fuzzy Integrity Constraints in Relational Databases. In: *Proceedings of the 6th IFSA World Conggress*, São Paulo, Brazil, 389-392.

Patig, S. (2006) Evolution of Entity-Relationship Modeling. *Data & Knowledge Engineering* **56** (2), 122-138.

Raju, K.V.S.V.N. and Majumdar, A.K. (1988) Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems* **13** (2), 129-166.

Savnik, I. and Flach, P.A. (2002) Discovery of Multi-valued Dependencies from Relations. *Intelligent Data Analysis Journal* **4** (3-4), 195-211.

Saxena, P.C. and Tyagu, B.K. (1995) Fuzzy functional dependencies and independencies in extended fuzzy relational database models. *Fuzzy Sets and Systems* **69** (1), 65-89.

Shenoi, S., Melton, A. and Tan, L.T. (1992) Functional dependencies and normal norms in the fuzzy relational database model. *Information Sciences* **60**, 1-28.

The Insurance Company (2000)© Sentient Machine Research, http://www.smr.nl.

Ullman, J.D. and Widom, J. (1997) *A First Course in Database Systems*. Prentice Hall, Inc., a Simon & Schuster Company.

Ullman, J.D. (1988) *Principles of Database and Knowledge-Based Systems*. Maryland, Computer Sciences Press Inc.

Wang, S.L., Shen, J.W. and Hong, T.P. (2002) Incremental discovery of functional dependencies based on partitions. *Intelligent Data Analysis*.

Wei, Q. and Chen, G.Q. (2003a) An Efficient Algorithm on Mining a Minimal Set of Functional Dependencies with Degrees of Satisfaction. *International Conference of IFSA 2003*, Istanbul, Turkey. Springer Verlag, Berlin.

Wei, Q. and Chen, G.Q. (2003b) Mining a Minimal Set of Functional Dependencies with Degrees of Satisfaction. *International Conference FIP 2003*, Beijing, China, March 1-4. Tsinghua University Press.

Wei, Q. and Chen, G.Q. (2004) Efficient Discovery of Functional Dependencies with Degrees of Satisfaction. *J. of Intelligent Systems* **19**, 1089-1110.

Wei, Q. and Chen, G.Q. (2006) Optimized Algorithm of Discovering Functional Dependencies with Degrees of Satisfaction. In: D. Ruan, et al., eds., *"Applied Artificial Intelligence". Proceedings of the 7th International FLINS Conference*, Word Scientific Press, 169-176.

Wei, Q., Chen, G.Q. and Kerre, E.E. (2002) Mining Functional Dependencies with Degrees of Satisfaction in Databases. In: *Proceedings of Joint Conference on Information Sciences*, Durham, NC, USA. Association for Intelligent Machinery, Inc.

Wijsen, J.Ng.R.T. and Calders, T. (1999) Discovering Roll-Up Dependencies. *Conference on Knowledge Discovery in Data. Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, California, United States. ACM Press, 213-222.

Wyss, C., Giannella, C. and Robertson, E. (2001) FastFDs: A heuristic-driven depth-first algorithm for mining functional dependencies from relation instances. Technical Report 551, CS Department, Indiana University, July.

Yang, Y.P. and Singhal, M. (1999) Fuzzy Functional Dependencies and Fuzzy Association Rules. *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, **LCNS 1676**, Springer Verlag, Berlin, 229-240.