On the Complexity of Mining Temporal Trends

Jef Wijsen and Robert Meersman
Vrije Universiteit Brussel

May 6, 1997

TR-16

A TIMECENTER Technical Report

Title	On the Complexity of Mining Temporal Trends
	Copyright © 1997 Jef Wijsen and Robert Meersman Vrije Universiteit Brussel. All rights reserved.
Author(s)	Jef Wijsen and Robert Meersman Vrije Universiteit Brussel
Publication History	May 1997. Paper at the SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97). May 1997. A TIMECENTER Technical Report.

TIMECENTER Participants

Aalborg University, Denmark
Christian S. Jensen (codirector)
Michael H. Böhlen
Renato Busatto
Heidi Gregersen
Kristian Torp

University of Arizona, USA
Richard T. Snodgrass (codirector)
Anindya Datta
Sudha Ram

Individual participants
Curtis E. Dyreson, James Cook University, Australia
Kwang W. Nam, Chungbuk National University, Korea
Keun H. Ryu, Chungbuk National University, Korea
Michael D. Soo, University of South Florida, USA
Andreas Steiner, ETH Zurich, Switzerland
Vassilis Tsotras, Polytechnic University, New York, USA
Jef Wijsen, Vrije Universiteit Brussel, Belgium

Any software made available via TIMECENTER *is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranty of merchantability and fitness for a particular purpose.*

The TIMECENTER icon on the cover combines two "arrows." These "arrows" are letters in the so-called *Rune* alphabet used one millennium ago by the Vikings, as well as by their precedessors and successors, The Rune alphabet (second phase) has 16 letters. They all have angular shapes and lack horizontal lines because the primary storage medium was wood. However, runes may also be found on jewelry, tools, and weapons. Runes were perceived by many as having magic, hidden powers.

The two Rune arrows in the icon denote "T" and "C," respectively.

Abstract

We investigate the computational complexity of mining certain trends in temporal databases. A simple example of such trend might be *"In general, salaries of employees do not decrease."* The trends considered are formalized by the construct of *trend dependency* (TD). TD's can compare attributes over time by using operators of $\{<, =, >, \leq, \geq \neq\}$. TD satisfaction is characterized by a *support* and *confidence*. As TD's may express meaningful trends, mining them is significant. The TD mining problem studied is the following task: Given a temporal database, find the TD of a specified form that holds with the highest confidence and with support greater than or equal to a specified minimum threshold. This problem is called TDMINE. Unlike most other work in data mining, we primarily focus on the computational complexity of the TDMINE problem—rather than on the performance of algorithms to solve it. Both the number of tuples (cardinality) and the number of attributes can be taken as the "size" of TDMINE. TDMINE can be solved in $\mathcal{O}(C^2)$ time where C is the cardinality. If the time requirements are expressed in function of the number of attributes—rather than the cardinality—then the problem turns out **NP**-complete. We discuss the practical implications of this result.

1 Introduction

Recently, there has been a growing interest in the mining of different types of *association rules* from large relational tables. These associations generally compare certain attribute values of a tuple with specified constants. An example is *"If degree is PhD then salary is large"* [YC96]. In this paper, we introduce rules (or, *dependencies*) that compare attributes *between* two tuples using operators of $\{<, =, >, \leq, \geq \neq\}$. An example might be *"The higher the degree, the higher the salary."* This type of dependencies turns out very useful in temporal databases as it allows capturing meaningful trends. Nearly all temporal dependencies that have so far been proposed in the literature compare attributes for equality only [JSS96], which greatly limits their expressivity.

For example, consider the relational schema $\{$*SS#, Rank, Sal*$\}$. A tuple $\{$*SS#:x, Rank:y, Sal:z*$\}$ means that the employee with the social security number x has rank y and salary z. Ranks are numbers between 1 and 5. Assume that I_1 and I_2 are two relations over this schema at two time points t_1 and t_2 respectively, where t_1 is earlier than t_2. Consider the regularity:

"If an employee's rank increases then his/her salary does not decrease."

which may be denoted:

$$(SS\#, =)(Rank, <) \circ (Sal, \leq)$$

and called a *trend dependency* (TD). The meaning is as follows: Let s_1 and s_2 be two tuples of I_1 and I_2 respectively with $s_1(SS\#) = s_2(SS\#)$—i.e., s_1 and s_2 represent the same employee—and $s_1(Rank) < s_2(Rank)$. Then our confidence in the reliability of the trend increases if $s_1(Sal) \leq s_2(Sal)$. On the other hand, $s_1(Sal) > s_2(Sal)$ would give evidence against the trend. So a specified TD can hold to a certain degree. The degree of TD satisfaction will be characterized by the common notions of *support* and *confidence*.

As TD's may express significant knowledge about the stored data, discovering them is interesting and important. Until now, most research in temporal database mining has addressed the problem of searching for sequences that are similar to a given target sequence [CHY96]. The aim of this study is to explore the complexity of discovering (or *mining*) TD's in temporal relations. The TD mining problem studied in this paper is the following task: Given a temporal database, choose among the TD's that obey a specified format, the TD with the highest confidence and with support greater than a specified minimum threshold. This problem is called TDMINE. A precise characterization will be given in the technical treatment later on.

Most work in data mining concerns in the first place the performance of algorithms. Some examples are [AS94, FL95, PCY95, SON95]. In this study, we proceed in a different way and start with analyzing the complexity of the TDMINE problem itself—rather than algorithms to solve it. The rationale behind this approach is that complexity analysis gives us important indications about the tractability of the problem in hand, which may complement algorithm design techniques.

The outline of the paper is as follows. The next section discusses some related work. The notion of TD is formalized in section 3. Section 4 gives a precise formulation of the data mining problems explored. The complexity results are given in section 5. Perhaps the most important contribution of this paper is the theorem that TDMINE is **NP**-complete if the input is characterized by the number of attributes involved in a TD. Some practical implications of the results are discussed in section 6. Finally, section 7 contains concluding remarks and open problems.

A final remark: In this paper, a temporal relation is limited to a pair (I_1, I_2) of classical relations. One may wonder why we only consider two time points, rather than a (possibly infinite) sequence of time points. In fact, the results presented can be extended to infinite time series of relations. However, we found out that the reduced formalism with two time instances has in it the full complexity of the data mining problem we are going to explore, while it considerably simplifies the technical treatment.

2 Related Work

Lately and independently of data mining, there has been a growing interest in dependencies for temporal databases [JSS96, WBBJ97]. All temporal dependencies found in the extensive overview of Jensen et al. [JSS96] compare attributes by using equality only. Our TD's compare attributes by operators of $\{<, =, >, \leq, \geq \neq\}$; they generalize the *dynamic functional dependencies* proposed by Wijsen [Wij95].

Association rules can take different forms [HF95, SA96]. Most work in association mining has concentrated on discovering rules of the form

$$p_1(A_1) \text{ and } p_2(A_2) \text{ and } \ldots p_n(A_n) \Rightarrow p_{n+1}(A_{n+1})$$

where $A_1, \ldots, A_{n+1}$ are all distinct attributes. Each $p_i(A_i)$ is an equation associating attribute A_i with a single value or a range of values of its domain. An example given by Srikant and Agrawal [SA96] is:

$$(Age : 30..49) \text{ and } (Married : Yes) \Rightarrow (NumCars : 2)$$

expressing that married people between 30 and 49 years old have two cars. The support s of an association rule is the percentage of tuples satisfying both the left-hand and the right-hand side of the rule.[1] The confidence is c if $c\%$ of the tuples satisfying the left-hand side of the rule also satisfy the right-hand side. Certain studies limit the length n of the rule to enable *visualization*. Fukuda et al. [FMMT96] deal with two-dimensional rules only (i.e., $n = 2$).

Note that the association rules just mentioned compare certain attribute values of a tuple with specified constants. Each individual tuple can give evidence for or against the association. The TD's proposed in this paper compare attributes in one tuple with the corresponding attributes in another tuple. That is, TD satisfaction is expressed in terms of tuple pairs—rather than individual tuples. Following the terminology of Baudinet et al. [BCW95], TD's are constraint-generating 2-dependencies, whereas classical association rules are constraint-generating 1-dependencies.

Before we turn to the technical treatment, we mention that TD's can be easily extended to capture data regularities within a single state (called intrastate dependencies in [JSS96]) or among more than two states. However, the restricted formalism is sufficiently powerful to show the full complexity of the data mining problem we are interested in.

3 Formalizing Trends

In this section we formalize the notion of *trend dependency* (TD) and *TD class*. To help understanding, we start with a simple example.

3.1 Illustrative Example

Consider the relations I_1 and I_2 shown in figure 1. Intuitively we think of I_1 as the tuples valid at some time t_1, and I_2 as the tuples valid at some later time t_2. The TD

$$\sigma = (SS\#, =)(Rank, <) \circ (Sal, \leq)$$

expresses that an employee's salary does not decrease if his/her rank increases. Clearly, employee $D4$ gives evidence against the trend: The rank of employee $D4$ increased while his/her salary decreased. On the other hand, employees $A1$ and $B2$ support the trend, as both their rank and salary increased. Finally, employee $C3$

[1] By the left-hand side of the rule, we mean the rule part preceding $\Rightarrow$. The right-hand side is what follows $\Rightarrow$.

I_1 :

SS#	Rank	Sal
A1	1	100
B2	2	80
C3	3	140
D4	1	120

I_2 :

SS#	Rank	Sal
A1	2	110
B2	3	90
C3	2	130
D4	2	110

Figure 1: Example database.

provides no argument for or against the trend, as his/her rank did not increase. We are going to formalize the above observations. The expression $(SS\#, =)(Rank, <)$ is called a *pattern*, and is said to be *satisfied* by a *tuple pair* (s_1, s_2) of $I_1 \times I_2$ if and only if $s_1(SS\#) = s_2(SS\#)$ and $s_1(Rank) < s_2(Rank)$. Likewise, the right-hand pattern $(Sal, \leq)$ is satisfied by the tuple pair (s_1, s_2) of $I_1 \times I_2$ if and only if $s_1(Sal) \leq s_2(Sal)$. So satisfaction is expressed in terms of tuple pairs of $I_1 \times I_2$. In figure 1, the cardinality of $I_1 \times I_2$ is 16. Every tuple pair satisfying the left-hand pattern $(SS\#, =)(Rank, <)$ gives evidence for or against the TD in hand, depending on whether or not it satisfies the right-hand pattern $(Sal, \leq)$. The confidence c is obtained by the number of tuple pairs satisfying both patterns divided by the number of tuple pairs satisfying the left-hand pattern; in the example $c = 2/3$. The support s is the number of tuple pairs satisfying both patterns divided by the total number of tuple pairs; in the example $s = 2/16$. So the TD σ is satisfied with support $2/16$, and with confidence $2/3$.

3.2 Trend Dependency

We are now going to define the notion of *trend dependency* (TD). Let $i, j \in \mathbb{N}$. We write $[i..j]$ for the set $\{i, i+1, \ldots, j\}$ and we write $(0, 1)$ for the smallest set containing every real number between 0 and 1. The number of elements in a set S is denoted $|S|$.

Definition. We assume the existence of a totally ordered set $(\mathbf{dom}, \leq)$. We define the set $\mathbf{op} = \{<, =, >, \leq, \geq \neq\}$ of *operators*, with their natural meaning. We assume the existence of a set $\mathbf{att}$ of *attributes*. Let $U \subseteq \mathbf{att}$. A *tuple* over U is a total function from U to $\mathbf{dom}$. A *relation* over U is a set of tuples over U. A *tuple pair* over U is a pair (s_1, s_2) of tuples over U. A *relation pair* over U is a pair (I_1, I_2) of relations over U.

For simplicity, we assume that all attributes of a tuple take their values from the same "domain" $\mathbf{dom}$. This limitation does not affect the results presented in this study in any significant way.

A *pattern* over U is a set $\{(A_1, \theta_1), \ldots, (A_m, \theta_m)\}$ where $A_1, \ldots, A_m$ are all distinct attributes of U, and $\theta_1, \ldots \theta_m$ are operators of $\mathbf{op}$. That is, a pattern over U is a (partial) function from U to $\mathbf{op}$. The domain of a pattern π is denoted $[\![\pi]\!]$.

$\{(A_1, \theta_1), (A_2, \theta_2), \ldots, (A_m, \theta_m)\}$ will be denoted $(A_1, \theta_1)(A_2, \theta_2) \ldots (A_m, \theta_m)$.

The tuple pair (s_1, s_2) over U is said to *satisfy* the pattern $(A_1, \theta_1) \ldots (A_m, \theta_m)$ iff $s_1(A_i)\ \theta_i\ s_2(A_i)$ for every $i \in [1..m]$.

A *trend dependency* (TD) over U is a statement $\pi \circ \rho$ where π and ρ are patterns over U. π and ρ are called the *left-hand* and the *right-hand* pattern of $\pi \circ \rho$ respectively. The semantics of TD's is now defined relative to a relation pair (I_1, I_2):

Let σ be a TD over U. Let (I_1, I_2) be a relation pair over U.
Let p be the number of tuple pairs of $I_1 \times I_2$. Let l be the number of tuple pairs of $I_1 \times I_2$ satisfying the left-hand pattern of σ. Let b be the number of tuple pairs of $I_1 \times I_2$ satisfying both the left-hand and the right-hand pattern

of σ. Let

$$s = \begin{cases} b/p & \text{if } p \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and } c = \begin{cases} b/l & \text{if } l \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then σ is said to be *satisfied* by (I_1, I_2) with *support* s and *confidence* c, denoted $(I_1, I_2) \models_c^s \sigma$. If $(I_1, I_2) \models_c^s \sigma$, then we also say that s and c are the support and the confidence of σ respectively (where (I_1, I_2) is implicitly understood). Clearly $0 \leq s \leq c \leq 1$. □

We now define the notion of *TD class*. A *TD class* determines a set of TD's by specifying the domain of the left-hand and the right-hand pattern, and by limiting the operators that can be used.

Definition. Let Θ be a nonempty subset of $\mathbf{op}$. Let X and Y be sets of attributes (i.e., $X, Y \subseteq \mathbf{att}$).
The *TD class* determined by X, Y and Θ (in that order), denoted $\mathcal{TD}_\Theta^{X \circ Y}$, is the smallest set containing the TD $\pi \circ \rho$ if the following is true:

- $[\![\pi]\!] = X$ and for every $A \in X$, $\pi(A) \in \Theta$, and
- $[\![\rho]\!] = Y$ and for every $A \in Y$, $\rho(A) \in \Theta$.

We write $\mathcal{TD}^{X \circ Y}$ as a shorthand for $\mathcal{TD}_{\mathbf{op}}^{X \circ Y}$. □

For example, $\mathcal{TD}_{\{<,>\}}^{\{A,B\} \circ \{C\}}$ contains all TD's
$(A, \theta_1)(B, \theta_2) \circ (C, \theta_3)$ where each of θ_1, θ_2 and θ_3 is either " $<$ " or " $>$ ". An example element is $(A, <)(B, >) \circ (C, <)$.

4 TD Mining Problems

The data mining problem studied in this paper is the following task: Given a relation pair (I_1, I_2), select from a specified TD class the TD that is satisfied with the highest confidence and with support greater than or equal to some specified minimum threshold.

Definition. Let (I_1, I_2) be a relation pair over the set U of attributes. Let $X, Y \subseteq U$. Let Θ be a nonempty subset of $\mathbf{op}$. Given a minimum threshold $\tau \in (0, 1)$:
Determine TD $\sigma \in \mathcal{TD}_\Theta^{X \circ Y}$ for which the following is true (let $s, c \in (0, 1)$ such that $(I_1, I_2) \models_c^s \sigma$):

- $s \geq \tau$ *(threshold support)*, and
- For every $\sigma' \in \mathcal{TD}_\Theta^{X \circ Y}$ the following is true:
  If $(I_1, I_2) \models_{c'}^{s'} \sigma'$ with $s' \geq \tau$, then $c' \leq c$ *(maximal confidence)*.

This problem is called TDMINE$_\Theta$. TDMINE is a shorthand for TDMINE$_{\mathbf{op}}$. □

TDMINE$_\Theta$ is an optimization problem: Given a relation pair (I_1, I_2) and a TD class $\mathcal{TD}_\Theta^{X \circ Y}$, the task is to assign operators of Θ to attributes of $X \cup Y$ so as to maximize the confidence of the resulting TD. Moreover, the support of the desired TD must exceed a specified minimum threshold. An alternative problem would be to fix the confidence and maximize the support.

The first step in the analysis of algorithms and complexity is to find some parameter(s) characterizing the size of the input. What is the size of TDMINE? Probably the most obvious answer is to characterize the input by the number of tuples (cardinality). Alternatively, one could consider characterizing the input by the number of attributes (degree, or dimension). In the remainder of this paper, we will adopt the following conventions:

- Given a relation pair (I_1, I_2), we use C to denote the average number of tuples in each state. That is, $C = \frac{|I_1| + |I_2|}{2}$.
- Given a TD class $\mathcal{TD}_\Theta^{X \circ Y}$, we use D to denote the number of attributes involved. That is, $D = |X \cup Y|$.

- Mostly a relation pair (I_1, I_2) and a TD class $\mathcal{TD}_{\Theta}^{X \circ Y}$ are implicitly understood.

Obviously, TDMINE$_\Theta$ can be solved in a brute force manner by an exhaustive algorithm that tries all possible combinations of operators, and computes the support and the confidence for each TD so obtained. The number of possible combinations is $\mathcal{O}(|\Theta|^D)$. Obviously, computing the confidence and the support of a given TD can be done in $\mathcal{O}(C^2)$ time.

Time proportional to C^2 and $|\Theta|^D$ is quite expensive. First consider $\mathcal{O}(C^2)$ time. Computing the confidence and the support of a specified TD requires comparing every tuple of I_1 with every tuple of I_2, resulting in $|I_1| \times |I_2|$ or $\mathcal{O}(C^2)$ comparisons. There seems to be no means to reduce the quadratic growth rate. Of course, practical performance improvements can be made by reducing the average cardinality C by "approximating" (I_1, I_2) by relation pairs with fewer tuples. Techniques that may be applicable are *generalization* [HCC93] and *sampling*. We will come back to this in section 6.

Next consider $\mathcal{O}(|\Theta|^D)$ time. The exponential growth rate is important if TD mining is done for relations with a fairly high number of attributes. An intriguing question is to which extent the naive exponential $\mathcal{O}(|\Theta|^D)$ algorithm can be improved. Interestingly, we are going to show that TDMINE is **NP**-complete and can be solved in polynomial time only if **P**=**NP**. As we indicated before, TDMINE is an optimization problem. For studying its complexity, we transform it into a roughly equivalent decision problem, called TDMINE(D):

Definition. Let (I_1, I_2) be a relation pair over the set U of attributes. Let $X, Y \subseteq U$. Let Θ be a nonempty subset of $\mathbf{op}$. Given $s', c' \in (0, 1)$:

Determine whether for some $\sigma \in \mathcal{TD}_{\Theta}^{X \circ Y}$ holds: $(I_1, I_2) \models_c^s \sigma$ with $s \geq s'$ and $c \geq c'$.

This problem is called TDMINE(D)$_\Theta$. TDMINE(D) is a shorthand for TDMINE(D)$_{\mathbf{op}}$. □

Hence, TDMINE(D)$_\Theta$ is the following problem: Determine whether there exists a TD in a specified TD class with confidence and support above given thresholds. Obviously, TDMINE$_\Theta$ is at least as hard as TDMINE(D)$_\Theta$: If we have a polynomial-time algorithm for TDMINE$_\Theta$, then we certainly do for TDMINE(D)$_\Theta$. However, it turns out that TDMINE(D)$_\Theta$ is **NP**-complete for certain Θ, as we are going to show.

5 Complexity Results

In this section we are going to explore the complexity of TDMINE(D). This leads to the following interesting results:

- TDMINE(D) is **NP**-complete.
- TDMINE(D)$_{\{<,=,>\}}$, on the other hand, is in **P**.

A polynomial time algorithm for the latter problem is given at the end of this section. Some practical implications of these complexity results are discussed in the next section.

5.1 TDMINE(D) is NP-complete

Lemma 1 *If* $(I_1, I_2) \models_c^s \pi \circ (A, \theta)$ *with* $\theta \in \mathbf{op}$ *then for some* $s' \geq s$*, for some* $c' \geq c$*,* $(I_1, I_2) \models_{c'}^{s'} \pi \circ (A, \theta')$ *with* $\theta' \in \{\leq, \geq, \neq\}$.

Proof. Trivial. □

Theorem 1 *TDMINE(D) is* **NP***-complete.*

Proof. TDMINE(D) can be very easily solved by a *nondeterministic* polynomial algorithm, one that guesses a TD of the specified TD class and computes the support and confidence in polynomial time; hence TDMINE(D) is in **NP**. We are now going to prove that 3SAT can be reduced to TDMINE(D).

	x_1	x_2	x_3	...	x_v	r	
I_1 :	.5	.5	.5	...	.5	.5	
I_{21} :	0	.5	.5	...	.5	.5	
	.5	0	.5	...	.5	.5	
	.5	.5	0	...	.5	.5	
				...			
	.5	.5	.5	...	0	.5	
	1	.5	.5	...	.5	.5	
	.5	1	.5	...	.5	.5	
	.5	.5	1	...	.5	.5	
				...			
	.5	.5	.5	...	1	.5	
I_{22} :	0	0	1	...	.5	0	3 tuples
	0	0	1	...	.5	.5	corresponding to
	0	0	1	...	.5	1	$\neg x_1 \vee \neg x_2 \vee x_3$
				...			

Figure 2: Construction example.

Consider the propositional formula Δ:

$$\bigwedge_{i=1..m} \tau_{i1} \vee \tau_{i2} \vee \tau_{i3}$$

where τ_{ij} is either a variable or the negation of one. Let V be the smallest set containing each variable appearing in Δ. Let v be the number of elements in V. $v \geq 3$ is assumed without loss of generality. Let U be a set of attributes. For convenience, we assume $U = V \cup \{r\}$ where $r \notin V$. We describe the reduction R from 3SAT to TDMINE(D) next. Let $x \in U$. We write $t_{x=a}$ for the tuple t over U satisfying: $t(x) = a$, and $t(y) = 0.5$ if $y \neq x$. For every $i \in [1..m]$ we define three tuples, denoted t_{i0}, t_{i1} and t_{i2}, with for each $x \in U$, for each $j \in \{0, 1, 2\}$,

- $t_{ij}(x) = 0$ if $\tau_{i1} \vee \tau_{i2} \vee \tau_{i3}$ contains the negation of x,
- $t_{ij}(x) = 1$ if $\tau_{i1} \vee \tau_{i2} \vee \tau_{i3}$ contains x—without negation,
- $t_{ij}(r) = j/2$, and
- $t_{ij}(x) = 0.5$ otherwise.

No term of Δ contains both x and the negation of x, is assumed without loss of generality. Let I_{21} be the smallest relation over U containing $t_{x=0}$ and $t_{x=1}$ for every $x \in V$. Let I_{22} be the smallest relation over U containing t_{i0}, t_{i1} and t_{i2} for every $i \in [1..m]$. Let $I_2 = I_{21} \cup I_{22}$. Let I_1 be a singleton containing $t_{r=0.5}$. The construction is illustrated in figure 2. Let p be the number of tuple pairs of $I_1 \times I_2$. Let

$$s' = \frac{v}{p} \text{ and } c' = 1$$

Note that $s' > 0$. We claim that R is a reduction from 3SAT to TDMINE(D). To prove our claim, we have to establish two things: (1) that any formula Δ has a satisfying truth assignment iff for some $\pi \circ \rho \in \mathcal{TD}^{V \circ \{r\}}$ holds, $(I_1, I_2) \models_c^s \pi \circ \rho$ with $s \geq s'$ and $c \geq c'$; and (2) that R can be computed in space $\log n$.

Assume for some $\pi \circ \rho \in \mathcal{TD}^{V \circ \{r\}}$ holds, $(I_1, I_2) \models_c^s \pi \circ \rho$ with $s \geq s'$ and $c \geq c'$. By lemma 1, $\rho(r) \in \{\leq, \neq, \geq\}$ is assumed without loss of generality. We are going to show that Δ has a satisfying truth assignment. Let k be the number of tuple pairs of $I_1 \times I_{21}$ satisfying π. Obviously, $k \leq v$. Clearly, the number of tuple pairs of $I_1 \times I_{22}$ satisfying π is a multiple of three—let it be $3n$. Let κ (Greek lowercase kappa) be a number such that

$$\kappa = \begin{cases} 0 & \text{if } \rho(r) = \text{“} \neq \text{”} \\ k & \text{otherwise} \end{cases}$$

Symbol	Interpretation
π	left-hand pattern
ρ	right-hand pattern
$3n$	number of pairs of $I_1 \times I_{22}$ satisfying π
k	number of pairs of $I_1 \times I_{21}$ satisfying π
κ	number of pairs of $I_1 \times I_{21}$ satisfying both π and ρ
p	cardinality of $I_1 \times I_2$
r	single attribute in $[\![\rho]\!]$
v	number of variables

Figure 3: Symbol interpretation.

Then $(I_1, I_2) \models_c^s \pi \circ \rho$ with

$$s = \frac{\kappa + 2n}{p} \text{ and } c = \frac{\kappa + 2n}{k + 3n}$$

Note that *not* $\kappa = n = 0$ since $s \geq s' > 0$. Then $c \geq c'$ and $s \geq s'$ imply:

$$\kappa = k \text{ and } n = 0 \text{ and } k \geq v.$$

Hence, $\rho(r)$ is either " $\leq$ " or " $\geq$ ", and $k = v$. One can easily check that $k = v$ implies that $\pi \circ \rho$ is in $\mathcal{TD}^{V \circ \{r\}}_{\{\leq, \geq\}}$. Let $\{x_{i1}, x_{i2}, x_{i3}\}$ be the set of variables occurring in the i^{th} term of Δ. Let

$$\theta_{ij} = \begin{cases} \text{" } \leq \text{"} & \text{if } t_{i0}(x_{ij}) = 0 \\ \text{" } \geq \text{"} & \text{if } t_{i0}(x_{ij}) = 1 \end{cases} \quad (j = 1..3)$$

$n = 0$ implies that for each $i \in [1..m]$ the following is true:

$$\pi(x_{i1}) = \theta_{i1} \text{ or } \pi(x_{i2}) = \theta_{i2} \text{ or } \pi(x_{i3}) = \theta_{i3}$$

Obviously, if we identify " $\leq$ " and " $\geq$ " with **false** and **true** respectively, then π is a truth assignment satisfying Δ. Conversely, from the foregoing it is easy to see that if Δ has a satisfying truth assignment, then for some $\pi \circ \rho \in \mathcal{TD}^{V \circ \{r\}}$ holds, $(I_1, I_2) \models_c^s \pi \circ \rho$ with $s \geq s'$ and $c \geq c'$. To see that R can be computed in $\log n$ space, note that $R(\Delta)$ can be written directly from Δ. This concludes the proof. □

Remark that the TD mined in theorem 1 only uses the operators " $\leq$ " and " $\geq$ ". This leads to the following corollary.

Corollary 1 *If Θ contains both " $\leq$ " and " $\geq$ " then TDMINE(D)$_\Theta$ is* **NP***-complete.*

Proof. This follows immediately from the fact that in the proof of theorem 1, the TD $\pi \circ \rho$ belongs to $\mathcal{TD}^{V \circ \{r\}}_{\{\leq, \geq\}}$. □

5.2 TDMINE(D)$_{\{<,=,>\}}$ is in P

We first show that TDMINE(D)$_{\{<,=,>\}}$ is in **P**, and then we give a polynomial-time algorithm to solve TDMINE$_{\{<,=,>\}}$.

Theorem 2 *TDMINE(D)$_{\{<,=,>\}}$ is in* **P**.

Proof. For each (s_1, s_2) in $I_1 \times I_2$, one can construct in $\mathcal{O}(D)$ time the *unique* TD $\pi \circ \rho$ of $\mathcal{TD}^{X \circ Y}_{\{<,=,>\}}$ such that (s_1, s_2) satisfies both π and ρ, where $D = |X \cup Y|$. For each TD so constructed one can compute the support and confidence in polynomial time. Finally, one can select the TD with the highest confidence among the TD's of which the support exceeds the specified minimum threshold. □

Theorem 2 suggests a naive way to solve TDMINE$_{\{<,=,>\}}$. A better algorithm is presented in figure 4. The algorithm proceeds in three steps. During the *pattern phase*, we store for every (s_1, s_2) of $I_1 \times I_2$ the *unique* TD $\pi \circ \rho$ of $\mathcal{TD}^{X \circ Y}_{\{<,=,>\}}$ such that both π and ρ are satisfied by (s_1, s_2). This step requires $\mathcal{O}(C^2)$ time. The resulting list with $\mathcal{O}(C^2)$ tuples is sorted during the *sorting phase*. This can be done in $\mathcal{O}(C^2 \log C)$ time. Finally, the *statistics phase* computes the support and the confidence for each TD stored in the pattern phase, and returns the "strongest" TD. This step requires $\mathcal{O}(C^2)$ time. So the algorithm is $\mathcal{O}(C^2 \log C)$. When we characterize the input by the number of attributes, then the time requirements are $\mathcal{O}(D)$.

6 Discussion

It turns out that the proposed algorithms for solving TDMINE are at least $\mathcal{O}(C^2)$, where C is the average cardinality. This is no surprise, as computing the confidence of a specified TD requires comparing every tuple of I_1 with every tuple of I_2. Of course, practical performance improvements can be obtained by reducing C. Perhaps the most obvious way to reduce the number of tuples is by *sampling*. *Generalization* [HCC93] has also been used for reducing the cardinality of a relation. Another way to speed up the TD mining process might be to use visualization techniques, which are generally considered a useful method for discovering patterns in data sets. This, of course, relies on some user intervention. The problem here is that it is difficult to visualize data spaces with high dimensionality. Every attribute appearing in a TD is a dimension. Moreover, the time dimension is inherent in every TD. The practicality of visualization therefore depends on the possibility to solve a given TDMINE problem by only dealing with "short" TD's at a time.

So a question of practical importance is: Can a given TDMINE problem involving D attributes be solved by (a) first *decomposing* it in polynomial time into a number of TDMINE subproblems of a specified visualizable dimension (usually 2 or 3), (b) then solving the smaller subproblems using visualization techniques, and (c) finally merging the solutions in polynomial time so as to obtain a solution for the original TDMINE problem? It is of interest to note immediately that "shortening" the left-hand pattern of a TD can result in an increase as well as a decrease of the confidence. Recall that the confidence c of a TD $\pi \circ \rho$ is equal to b/l where b is the number of tuple pairs satisfying both π and ρ, and l is the number of tuple pairs satisfying π. Replacing π by a proper subset of π will result in both b and l increasing. So c can increase as well as decrease. More fundamentally, we showed that the TDMINE problem is **NP**-complete if the input is characterized by the number of attributes. In the suggested decomposition strategy, all subproblems are of the same specified, visualizable dimension and hence the time required to solve any one subproblem is irrespective of D. But then the decomposition strategy corresponds to a polynomial-time algorithm, which exists only if **P**=**NP**. This is a strong indication that there is no such decomposition strategy. As a consequence, visualization does not seem applicable to the discovery of TD's among a "large" (i.e., non-visualizable) number of attributes.

We showed that TDMINE is **NP**-complete if expressed as a function of the number of attributes D. Time proportional to $|\Theta|^D$ is expensive if D is large, i.e., if we are mining TD's involving a fairly large number of attributes. Hu and Cercone [HC96] describe *attribute reduction* techniques to eliminate attributes that are redundant and/or irrelevant to the knowledge discovery process they are studying. Although their approach is not directly applicable to our TD's, the idea is interesting and needs further attention.

TDMINE$_\Theta$ requires that the operators appearing in the outcome TD belong to Θ. One could consider specifying the set of allowed operators on an attribute by attribute basis—rather than for the TD as a whole. In particular, for certain attributes, such as $SS\#$, equality and inequality are often the only meaningful operators. We found that many meaningful TD's compare primary key attributes by "=". This can be explained as follows. Often tuples represent entities (for example, employees); and primary keys represent identifiers of entities (for example, social security number). In many cases, one is interested to see how certain properties (for example, salary) of an entity evolve in time. In these cases, one will typically look for TD's of which the left-hand pattern includes $(K_1, =), \ldots, (K_n, =)$ where $\{K_1, \ldots, K_n\}$ is the primary key of the relational schema under consideration. This gives rise to a variant of TDMINE$_\Theta$ where certain operators are fixed. A possible instance of this new mining problem might be: Choose θ_1 and θ_2 so as to optimize the confidence of $(SS\#, =)(Rank, \theta_1) \circ (Sal, \theta_2)$. Major performance improvements can be made for mining such TD's: Let σ be a TD of which the left-hand pattern contains $(K_1, =), \ldots, (K_n, =)$ where $\{K_1, \ldots, K_n\}$ is the primary key. One can easily check that if the relations I_1 and I_2 are kept ordered by the primary key then the support and confidence of σ can be computed in $\mathcal{O}(C)$ time.

Algorithm 1 *Algorithm to solve TDMINE*$_{\{<,=,>\}}$.

Input: A relation pair (I_1, I_2) over U,
$i = |I_1| \times |I_2| \geq 1$,
two sets $X, Y \subseteq U$,
a minimum threshold support $\tau \in (0, 1)$.
Output: An answer to TDMINE$_{\{<,=,>\}}$.

∗PATTERN PHASE∗
make `anArray` empty
FOR `s` in I_1 *LOOP*
  FOR `t` in I_2 *LOOP*
    FOR `A` in $X \cup Y$ *LOOP*
      IF `s(A)` $<$ `t(A)` *THEN* `q(A)` := " $<$ "
      ELSIF `s(A)` $=$ `t(A)` *THEN* `q(A)` := " $=$ "
      ELSE `q(A)` := " $>$ "
      END-IF
    END-LOOP
    add `q` to `anArray`
  END-LOOP
END-LOOP
∗SORTING PHASE∗
sort `anArray` by X, Y
add "sentinel" at the end of `anArray`
∗STATISTICS PHASE∗
`maxc`:=0; `maxTD`:=a dummy pattern
`l1`:=1; `l2`:=1; `r1`:=1; `r2`:=1
WHILE `l1` $\leq i$ *LOOP*
  `x`:=`anArray(l1)[`X`]`
  WHILE `anArray(l2)[`X`]`=`x` *LOOP*
    `l2`:=`l2`+1
  END-LOOP
  `l` := `l2` − `l1`
  WHILE `r1` $<$ `l2` *LOOP*
    `y`:=`anArray(r1)[`Y`]`
    WHILE `anArray(r2)[`Y`]`=`y` and `r2` $<$ `l2` *LOOP*
      `r2`:=`r2`+1
    END-LOOP
    `r` := `r2` − `r1`
    `s` := `r`$/i$; `c` := `r`$/$`l`
    IF `s` $\geq \tau$ and `c` $>$ `maxc` *THEN*
      `maxc`:=`c`; `maxTD`:=`anArray(r1)[`$X \cup Y$`]`
    END-IF
    `r1`:=`r2`
  END-LOOP
  `l1`:=`l2`
END-LOOP
RETURN `maxc`, `maxTD` □

Figure 4: Algorithm to solve TDMINE$_{\{<,=,>\}}$.

7 Summary and Open Problems

We introduced *trend dependencies* (TD's), which allow capturing significant temporal trends. We studied the problem TDMINE: Given a temporal databases, mine the TD of a specified TD class with the highest confidence and with support greater than or equal to a specified minimum threshold. Time requirements were expressed in terms of the average cardinality C and the number of attributes D. We showed that TDMINE is **NP**-complete: If time requirements are expressed as a function of D then TDMINE can be solved in polynomial time only if **P**=**NP**. We argued that TDMINE cannot be "scaled down" to lower dimensions, which limits the practicality of visualization techniques for discovering TD's among many attributes. Although solving TDMINE is generally expensive, we indicated some special cases with reduced time requirements. Further research is required to study and apply techniques for tuple and attribute reduction.

References

[AS94] A. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, 1994.

[BCW95] M. Baudinet, J. Chomicki, and P. Wolper. Constraint-generating dependencies. In *Proc. 5th Int. Conf. on Database Theory*, LNCS 893, pages 322–337. Springer-Verlag, 1995.

[CHY96] M.-S. Chen, J. Han, and P.S. Yu. Data mining: An overview from a database perspective. *IEEE Trans. on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[FL95] C. Faloutsos and K.I. Lin. Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 163–174, San Jose, CA, 1995.

[FMMT96] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 13–23, Montreal, Canada, 1996.

[HC96] X. Hu and N. Cercone. Mining knowledge rules from databases: A rough set approach. In *Int. Conf. Data Engineering*, pages 96–105, New Orleans, Louisiana, 1996.

[HCC93] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40, 1993.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. Int. Conf. Very Large Data Bases*, pages 420–431, Zürich, Switzerland, 1995.

[JSS96] C.S. Jensen, R.T. Snodgrass, and M.D. Soo. Extending existing dependency theory to temporal databases. *IEEE Trans. on Knowledge and Data Engineering*, 8(4):563–582, 1996.

[PCY95] J.S. Park, M.-S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 175–186, San Jose, CA, 1995.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 1–12, Montreal, Canada, 1996.

[SON95] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. Int. Conf. Very Large Data Bases*, pages 432–443, Zürich, Switzerland, 1995.

[WBBJ97] X.S. Wang, C. Bettini, A. Brodsky, and S. Jajodia. Logical design for temporal databases with multiple granularities. *to appear in ACM Trans. on Database Systems*, 22(2), 1997.

[Wij95] J. Wijsen. Design of temporal relational databases based on dynamic and temporal functional dependencies. In S. Clifford and A. Tuzhilin, editors, *Recent Advances in Temporal Databases*, Workshops in Computing, pages 61–76. Springer, 1995.

[YC96] S.J. Yen and A.L.P. Chen. The analysis of relationships in databases for rule derivation. *Journal of Intelligent Information Systems*, 7(3):235–259, 1996.