Fuzzy Calendar Algebra and Its Applications to Data Mining

Wan-Jui Lee
Department of Electrical Engineering,
National Sun Yat-Sen University, Taiwan
wrlee@water.ee.nsysu.edu.tw

Shie-Jue Lee
Department of Electrical Engineering,
National Sun Yat-Sen University, Taiwan
leesj@ee.nsysu.edu.tw

Abstract

Temporal expressions are widely used in our daily life. Calendar algebra has been studied for years to provide a formal specification for constructing temporal expressions. However, temporal requirements specified by human beings tend to be ill-defined or uncertain. To deal with such kind of uncertain information, we propose the fuzzy calendar algebra which allows users to describe desired temporal expressions easily and naturally. The operations provided reflect the way in which people reason about temporal requirements in daily life. By using the fuzzy calendar algebra, users can define complicated calendars with multiple time granularities in which different time intervals can have different weights according to their matching degrees to the specified calendar. This can help users to discover the knowledge in the time intervals that are of interest to them. We show the usefulness of the algebra by incorporating it with an incremental data miner to mine fuzzy temporal association rules from temporal databases.

1. Introduction

Efforts to construct and reason about temporal expressions have been conducted for a long time and have become an important topic in the areas of temporal databases, problem solving, natural language understanding, and data mining. Calendar algebra has been studied for years to provide a formal specification for constructing temporal expressions in terms of closely related granularities (e.g., day, month, year) that are organized into calendars (e.g., 3 February, 1978). Key elements, especially expressiveness, effectiveness, and compactness, have been investigated for calendar algebra. For expressiveness, the class of calendars represented in the formalism should be large enough to be of practical use. As for effectiveness, algorithms to reason about different time granularities should be provided in the framework. In particular, an effective solution should be provided to the problem of equivalence and classification which concern the identity of two calendar representations and the containment of one calendar to another, respectively. Finally, temporal expressions should be described as compact as possible.

The research of calendar algebra can be roughly categorized into two directions, the generation of temporal granularities [4, 6, 11, 5, 8, 9, 15] and the definition of operations for constructing calendars [7, 20, 16, 14]. Various operations have been defined to allow users to capture the characteristics of calendars naturally and expressively. Soo et al. proposed the uniform representation of calendar systems and tried to incorporate the idea into SQL2 [18]. A mechanism supporting the definition of calendars and a query language for them were developed. Three kinds of data types, event, interval, and span, are defined in the query language. In [7], another calendar system that supports calendars in a database system with extensible relational and active features was proposed. A language was developed to support the specification of a calendar, and operations like dicing and slicing were defined for creating desired time intervals. Other operations, such as merge, select, and extract, were also proposed in [20]. Features about calendar operations were also studied. Ramaswamy et al. [16] used calendar algebra to select interesting time periods in which knowledge could be discovered. Three operations, always, sometimes, and temporal, were introduced to specify queries in the object-oriented context in [14]. These operations are to be used to evaluate a proposition or answer questions, e.g., whether an event always/sometimes occurs in a time period, or when a certain event holds. However, temporal requirements specified by human beings tend to be ill-defined or uncertain. To deal with such kind of uncertain information, we borrow the fuzzy set theory [19] and propose the fuzzy calendar algebra to allow users to describe desired temporal expressions easily and naturally in terms of fuzzy calendars.

Five operations, **and**, **or**, **not**, **xor**, and **sub**, are provided to help users construct and manipulate complicated fuzzy calendars. These operations reflect the way in which people reason about temporal requirements in daily life. By using the fuzzy calendar algebra, users can define complicated calendars with multiple time granularities in which different time intervals can have different weights according to their matching degrees to the specified calendar. This can help users to discover the knowledge in the time intervals that are of interest to them. We show the usefulness of the algebra by incorporating it with an incremental data miner to mine fuzzy temporal association rules from temporal databases. Firstly, the transactions of the database are divided into a sequence of partitions. Then the weight of each partition is inferred, denoting the matching degree of the corresponding time period to the fuzzy calendar specified in the user's queries. Temporal patterns in the divided time periods of different matching degrees are then efficiently produced, and the number of database scans can be effectively reduced. Experimental results have shown that this approach of finding fuzzy temporal association rules is very effective.

The rest of the paper is organized as follows. In Section 2, the fuzzy calendar algebra, together with the construction of fuzzy calendars, is described. Operations for the fuzzy calendar algebra are defined. A formal description about fuzzy temporal association rules is given and the process of discovering fuzzy temporal association rules from temporal databases is described in Section 3. Simulation results are presented in Section 4. Finally, a conclusion is given in Section 5.

2. Fuzzy Calendar Algebra

A calendar is, in general, a structured collection of time intervals. Operations such as during, overlaps, meets, etc., have been proposed for users to construct their own calendars [16]. However, it is hard or even impossible for users to provide a crisp description about their desired calendars. To formulate human reasoning into the process of knowledge discovery, fuzzy theory is adopted for the construction of calendars in this section. Fuzzy concepts and operations are introduced to help users express their desired calendars easily and conveniently.

2.1. Basic Fuzzy Calendars

To construct a calendar, the hierarchy of time granularity, e.g. week, month, and year, has to be determined to handle descriptions of multiple time granu-

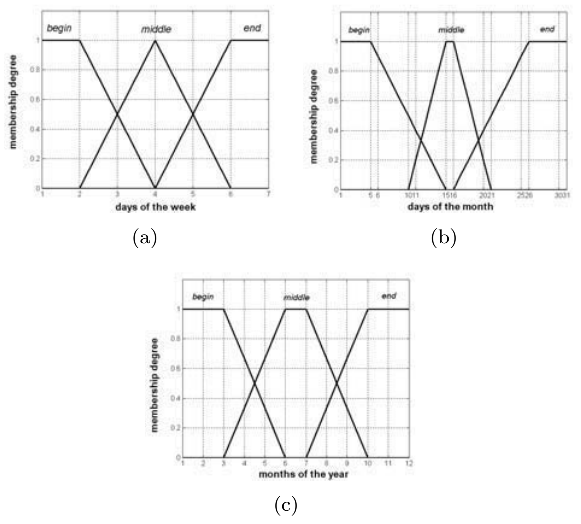

Figure 1. Basic fuzzy calendars associated with the time granularity of (a) week, (b) month, and (c) year.

larities [10]. For each time granularity, fuzzy sets which describe the distribution of all the time intervals in the time granularity can be specified. Each fuzzy description of a time granularity, e.g., *in the middle of a year*, *at the very beginning of a month*, or *at the end of a week*, etc., forms a *basic* fuzzy calendar.

Definition 1 *A* basic *fuzzy calendar, A, characterizes a fuzzy proposition about the collection of time intervals in a time granularity U, described by a membership function μ_A where*

$$\mu_A : T_i \rightarrow [0, 1]$$

for every time interval $T_i \in U$. The function value $\mu_A(T_i)$ indicates the matching degree of T_i to A.

Some example basic fuzzy calendars are shown in Figure 1. Usually, the shapes and the number of fuzzy sets describing a time granularity are arbitrarily specified by the user. Hedges such as *very* and *more or less* can also be used. With such fuzzy descriptions about time, users don't have to know the exact boundaries between interesting and non-interesting time intervals. Furthermore, the time intervals which are more important can have a larger membership degree and will contribute a bigger influence, which is intuitively desirable. Fuzzy calendars can also be used to describe crisp time intervals. For example, Wednesday and May can be described by fuzzy calendars with singletons, as shown in Figure 2. Temporal relations such as *before* and *after*

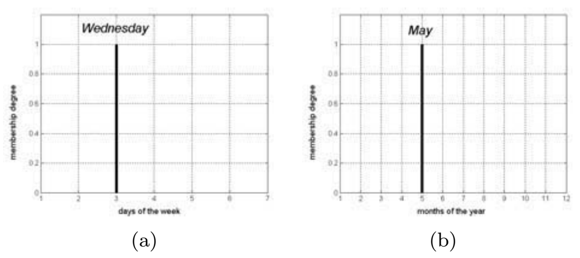

Figure 2. Singletons describing crisp time intervals, (a) Wednesday and (b) May, respectively.

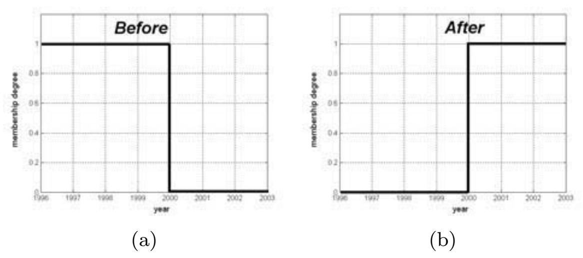

Figure 3. Membership functions describing the years (a) before and (b) after year 2000, respectively.

can also be represented by setting membership functions to be step functions, as shown in Figure 3.

2.2. Operations on Fuzzy Calendars

In real life, complicated temporal expressions, such as *on the weekends and at the end of a year*, are required and it is important for users to easily describe these temporal requirements. We provide five operations, **and**, **or**, **not**, **xor**, and **sub**, to help users to construct complex fuzzy calendars based on basic fuzzy calendars.

Definition 2 *A fuzzy calendar is defined recursively as follows:*

1. *A basic fuzzy calendar is a fuzzy calendar.*
2. *Let A and B be two fuzzy calendars. Then A **and** B, A **or** B, **not** A, A **xor** B, and A **sub** B are also fuzzy calendars.*

The semantics of these operations are described below.

Definition 3 *Let A and B be fuzzy calendars with membership functions μ_A and μ_B, respectively.*

1. *A **and** B denotes $A \wedge B$, with the resulting membership function $\mu_{A\wedge B}$ defined by*
$$\mu_{A\wedge B} \equiv \mu_A \times \mu_B. \tag{1}$$
2. *A **or** B denotes $A \vee B$, with the resulting membership function $\mu_{A\vee B}$ defined by*
$$\mu_{A\vee B} \equiv \mu_A + \mu_B - \mu_A \times \mu_B. \tag{2}$$
3. ***not** A denotes the complement of A, $\bar{A}$, and the resulting membership function $\mu_{\bar{A}}$ is defined by*
$$\mu_{\bar{A}} \equiv 1 - \mu_A. \tag{3}$$
4. *A **xor** B denotes the symmetric difference of A and B, $A \oplus B$, with the resulting membership function $\mu_{A\oplus B}$ defined by*
$$\begin{aligned}\mu_{A\oplus B} \equiv\ & \mu_A \times (1-\mu_B)^2 + \mu_B \times (1-\mu_A)^2 \\ & +\mu_A \times \mu_B \times (1 - \mu_A \times \mu_B).\end{aligned} \tag{4}$$
5. *A **sub** B denotes the subtraction of calendar B from calendar A, $A - B$, and*
$$\mu_{A-B} \equiv \mu_A - \mu_A \times \mu_B \tag{5}$$
is the resulting membership function.

Note that we use the algebraic sum and product, respectively, for the definitions of **or** and **and**, instead of *max* and *min* that are usually used. The main reason is that *max* and *min* are not satisfactory in some situations. For instance, when we take the **and** of two fuzzy sets, we want the larger fuzzy set to have a larger impact on the result. But if we use the *min* operation, the larger fuzzy set will have no impact at all. Similarly, the smaller fuzzy set will have no impact on the result if *max* is chosen as the operation for **or**.

Consider Figure 1. Let the basic fuzzy calendars *begin of a week*, *middle of a week*, *end of a week*, *begin of a month*, *middle of a month*, *end of a month*, *begin of a year*, *middle of a year*, and *end of a year* be represented by *bw*, *mw*, *ew*, *bm*, *mm*, *em*, *by*, *my*, and *ey*, with membership functions μ_{bw}, μ_{mw}, μ_{ew}, μ_{bm}, μ_{mm}, μ_{em}, μ_{by}, μ_{my} and μ_{ey}, respectively. Then the complex fuzzy calendar c_1 which is *((in the middle of a month **and** at the end of a year) **or** (at the end of a week **and** at the beginning of a year))* can be described by the following membership function:

$$\begin{aligned}\mu_{c_1} =\ & (\mu_{mm\wedge ey}) + (\mu_{ew\wedge by}) \\ & -(\mu_{mm\wedge ey})\times(\mu_{ew\wedge by}).\end{aligned} \tag{6}$$

Let c_2 be the fuzzy calendar *((**not** at the end of a year) **and** (in the middle of a year, **but not** at the end of a month))*. Then it can be described by

$$\mu_{c_2} = (\mu_{\overline{ey}})\times(\mu_{my-em}). \tag{7}$$

Finally, the fuzzy calendar c_3 which is *(at the beginning* ***or*** *end of a month,* ***but not*** *simultaneously)* can be described by

$$\mu_{c_3} \quad = \quad \mu_{bm \oplus em} \tag{8}$$

with the **xor** operation.

To calculate the matching degree of a time period, T, to a certain fuzzy calendar, A, we transform A to a fuzzy proposition of the following form

$$\text{The date is in } A. \tag{9}$$

Then the matching degree is computed as $\mu_A(T)$. For example, referring to Figure 1, the matching degree of the time period T = 2003/09/20 (Sat.) to the fuzzy calendar c_1 of Eq.(6) is computed as

$$\begin{aligned}\mu_{c_1}(T) &= (\mu_{mm}(20) \times \mu_{ey}(9)) + (\mu_{ew}(6) \times \mu_{by}(9)) \\ &\quad -((\mu_{mm}(20) \times \mu_{ey}(9)) \times (\mu_{ew}(6) \times \mu_{by}(9))) \\ &= (0.2 \times 0.67) + (1.0 \times 0.0) - ((0.2 \times 0.67) \\ &\quad \times (1.0 \times 0.0)) = 0.13. \end{aligned} \tag{10}$$

3. Application in Knowledge Discovery

We show how fuzzy calendar algebra can be used to help discover knowledge from temporal databases. We assume that all the transactions are placed in order in a temporal database. By incorporating fuzzy calendar algebra with a data miner, called PWM, an interesting type of knowledge, called fuzzy temporal association rules, can be efficiently discovered from temporal databases. The fuzzy temporal association rules discovered should have weighted support and confidence greater than the user-specified minimum support and confidence, respectively.

3.1. Fuzzy Temporal Association Rules

Let $\mathcal{I} = \{i_1, i_2, ..., i_m\}$ be a set of items. Let D be a temporal database of transactions where each transaction t is associated with an identifier TID, a time information t_t indicating the time when the transaction occurred, and a set of items t_c such that $t_c \subseteq \mathcal{I}$. Let D be divided into a sequence of n partitions, P_1, P_2, $\dots$, and P_n, each P_i containing a set of transactions occurring in the corresponding time period T_i with the duration being that of the smallest time granularity. Mining fuzzy temporal association rules in a database is to discover interesting patterns from the partitions divided in D, with corresponding time periods weighted by their matching degrees to the queried fuzzy calendar. The fuzzy temporal association rules discovered should have weighted support and confidence greater than the user-specified support and confidence thresholds, respectively.

Let *FC* be the specified fuzzy calendar, and w_i be the matching degree, or weight, of the time period T_i corresponding to partition P_i in the database D, where w_i is calculated as in Eq.(10). For a given itemset $I \subseteq \mathcal{I}$, a transaction t is said to *contain* I if and only if $I \subseteq t_c$. Let $|P_i(I)|$ be the number of transactions containing itemset I in partition P_i. The *weighted count* of an itemset I in D, denoted $\sigma_D(I)$, is defined to be

$$\sigma_D(I) = \sum_{k=1}^{n}(|P_k(I)| \times w_k). \tag{11}$$

We say that an itemset I is *frequent*, with respect to a *support threshold* of $s\%$ if

$$\sigma_D(I) \geq [\sum_{k=1}^{n}(|P_k| \times w_k)] \times s\% \tag{12}$$

where $|P_k|$ denotes the number of transactions in partition P_k. The term $[\sum_{k=1}^{n}(|P_k| \times w_k)] \times s\%$ is called the *weighted count threshold* of D. A fuzzy temporal association rule with respect to a fuzzy calendar, *FC*, is an implication of the form

$$X \Rightarrow^{FC} Y \tag{13}$$

where $X \subseteq \mathcal{I}$, $Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. As usual [1], X and Y are required to be non-empty and a rule thus contains at least two items. The association rule $X \Rightarrow^{FC} Y$ is said to have *weighted support* $s\%$ in the database D if

$$\sigma_D(X \cup Y) = [\sum_{k=1}^{n}(|P_k| \times w_k)] \times s\%. \tag{14}$$

For an association rule $X \Rightarrow^{FC} Y$, let

$$\sigma_D(X \cup Y)/\sigma_D(X) = c\%. \tag{15}$$

The rule is said to hold in D with *weighted confidence* $c\%$.

For a given pair of confidence and support thresholds, $c\%$ and $s\%$, and a given fuzzy calendar *FC*, the problem of mining fuzzy temporal association rules from the database D is to find out all the association rules that have weighted confidence and support greater than or equal to $c\%$ and $s\%$, respecitvely. As usual, two subproblems are involved. The first subproblem is to find, with respect to *FC*, all frequent itemsets in D. The second subproblem is, from the set of frequent itemsets, to find out all the association rules that have a weighted confidence value greater than or equal to $c\%$. Given a frequent itemset I, associated rules related to I are constructed as follows. Let I be decomposed into X and Y such that $X \cup Y = I$ and $X \cap Y = \emptyset$. $X \Rightarrow^{FC} Y$ is an association rule if $\sigma_D(I)/\sigma_D(X) \geq c\%$. Since the solution to the second subproblem is straightforward [1], major research efforts have been spent on the first subproblem, i.e., finding frequent itemsets. Therefore, our research mainly focuses on the first subproblem. Given the support threshold $s\%$, our mining system finds out the set

$$L = \left\{ X | X \subseteq \mathcal{I} \wedge \sigma_D(X) \geq [\sum_{k=1}^{n}(|P_k| \times w_k)] \times s\% \right\} \quad (16)$$

from the temporal database D. For convenience, an itemset that contains exactly k items is called a k-*itemset*. The set of frequent k-itemsets is commonly denoted by L_k. Note that the fuzzy temporal association rules defined above are identical to traditional association rules if the weights of all the time periods in D are set to the same value.

3.2. Mining Fuzzy Temporal Association Rules

In mining a time-variant database, traditional mining methods treat transactions in different time periods individually and process them with the same procedure without fully considering the time-variant characteristics of the items and transactions. Consequently, some interesting rules may not be discovered. We provide a method to remedy this disadvantage. The fuzzy calendar algebra is used to describe desired temporal requirements, and then PWM (Progressive Weighted Miner) [13] is adopted to discover fuzzy temporal association rules from a database.

In general, a database is too large to be held in main memory. Thus, the data mining techniques applied to very large databases have to be highly scalable for efficient execution. By partitioning a transaction database into n partitions, $P_1, P_2, \ldots, P_n$, PWM employs a progressive filtering scheme to generate the set of candidate itemsets for the database. One partition is considered at a time, and the cumulated information discovered in previous partitions, including a progressive candidate set of 2-itemsets, C_2, their partial occurrence counts, and the corresponding partial supports required, is carried over along to the subsequent partitions. When all the partitions have been processed, the set of candidate itemsets, C, for the database can be obtained.

For each frequent itemset I, there must exist some partition P_k, $1 \leq k \leq n$, such that I is frequent from partition P_k to P_n. Suppose E is a candidate 2-itemset

	Partition	Date	TID	Items
D	P_1	2003/09/15 (Mon.)	1	A,C,D,E,F
			2	B,D,F
			3	A,D,E
	P_2	2003/09/16 (Tue.)	4	A,B,D,E,F
			5	A,B,C,E,F
			6	B,F
	P_3	2003/09/17 (Wed.)	7	A,D,E,F
			8	A,B,D,F
			9	A,D,F

Table 1. A transaction database.

from P_k to P_t, $1 \leq k \leq t \leq n$. Let the current partition be P_{t+1}. If we detect that E is not frequent from P_k to P_{t+1}, then E can be deleted from C_2. If E is indeed frequent, it must be frequent in some later partition $P_{k'}$, $k' > k$, and we can add it to C_2 again with starting partition being $P_{k'}$. Therefore, after each partition, P_i, is processed, new partial frequent 2-itemsets may be added into C_2, with their starting partition and partial counts being recorded. Furthermore, the old itemsets in C_2 are checked if they are continually frequent from its starting partition to P_i. If a candidate 2-itemset is no longer partially frequent, this itemset is removed from C_2. After all partitions have been processed, the set of candidate 2-itemsets, C_2, which is close to the set of frequent 2-itemsets, is obtained. Based on C_2, all candidate k-itemsets, $k >= 3$, are generated. Finally, one database scan is applied to calculate the supports of all candidate itemsets and frequent itemsets are determined. Fuzzy temporal association rules can then be formed from the discovered frequent itemsets.

3.3. An Example

We give an example for illustration. Consider the database D, shown in Table 1, which contains 9 transactions and is segmented into three partitions, P_1, P_2, and P_3, each containing 3 transactions. Let the support and confidence thresholds be 40% and 75%, respectively. Assume that the queried fuzzy calendar is the fuzzy calendar c_1 given in Section 2, i.e., c_1 represents *((in the middle of a month* ***and*** *at the end of a year)* ***or*** *(at the end of a week* ***and*** *at the beginning of a year))*. The membership function of c_1 is shown in Eq.(6). Assume that μ_{mm}, μ_{ey}, μ_{ew}, and μ_{by} are the membership functions shown in Figure 1.

Firstly, we calculate the weights w_1, w_2, and w_3 for partitions P_1, P_2, and P_3, respectively. The time interval T_1 of P_1 is 2003/09/15 (Mon.). The membership degrees of T_1 with respect to μ_{mm}, μ_{ey}, μ_{ew}, and μ_{by}

P_1		
itemset	start	count
AD	1	1.34
AE	1	1.34
DE	1	1.34
DF	1	1.34

Table 2. Candidate 2-itemsets generated from partition P_1.

are $\mu_{mm}(15) = 1.0$, $\mu_{ey}(9) = 0.67$, $\mu_{ew}(1) = 0.0$, and $\mu_{by}(9) = 0.0$, respectively. Therefore, w_1 becomes

$$\begin{aligned} w_1 &= \mu_{mm}(15)\times\mu_{ey}(9) + \mu_{ew}(1)\times\mu_{by}(9) \\ &-((\mu_{mm}(15)\times\mu_{ey}(9))\times(\mu_{ew}(1)\times\mu_{by}(9))) \\ &= (1.0\times0.67) + (0.0\times0.0) - ((1.0\times0.67) \\ &\times(0.0\times0.0)) = 0.67. \end{aligned}$$

Similarly, we have T_2 of P_2 being 2003/09/16 (Tue.) and T_3 of P_3 being 2003/09/17 (Wed.), and

$$\begin{aligned} w_2 &= \mu_{mm}(16)\times\mu_{ey}(9) + \mu_{ew}(2)\times\mu_{by}(9) \\ &-((\mu_{mm}(16)\times\mu_{ey}(9))\times(\mu_{ew}(2)\times\mu_{by}(9))) \\ &= (1.0\times0.67) + (0.0\times0.0) - ((1.0\times0.67) \\ &\times(0.0\times0.0)) = 0.67. \\ w_3 &= \mu_{mm}(17)\times\mu_{ey}(9) + \mu_{ew}(3)\times\mu_{by}(9) \\ &-((\mu_{mm}(17)\times\mu_{ey}(9))\times(\mu_{ew}(3)\times\mu_{by}(9))) \\ &= (0.8\times0.67) + (0.0\times0.0) - ((0.8\times0.67) \\ &\times(0.0\times0.0)) = 0.536. \end{aligned}$$

Now we generate candidate 2-itemsets C_2 for D partition by partition.

- Processing of P_1. The weighted count threshold for P_1 is $0.4\times3\times0.67 = 0.804$. The accumulative weighted count threshold for P_1 is also 0.804. Among all the possible combinations of 2-itemsets in P_1, only AD, AE, DE, and DF have accumulative weighted counts greater than or equal to 0.804. Therefore, these 2-itemsets, together with their accumulative weighted counts (count) and starting partitions (start), are recorded as shown in Table 2.
- Processing of P_2. The weighted count threshold for P_2 is $m_2 = 0.4\times3\times0.67 = 0.804$. The accumulative weighted count threshold from partition P_1 to partition P_2 is $0.804 + 0.804 = 1.608$. We find that 5 new 2-itemsets, AB, AF, BE, BF, and EF in P_2 have weighted counts greater than m_2, so they are added to C_2. Furthermore, the old itemsets AD, AE, DE, and DF have their accumulative weighted counts greater than 1.608. Therefore, they are retained in C_2. The information kept after processing P_2 is shown in the left part of Table 3.

P_2			P_3		
itemset	start	count	itemset	start	count
AB	2	1.34	AB	2	1.87
AD	1	2.01	AD	1	3.61
AE	1	2.68	AE	1	3.21
AF	2	1.34	AF	2	2.94
BE	2	1.34	BF	2	2.54
BF	2	2.01	DE	1	2.54
DE	1	2.01	DF	1	3.61
DF	1	2.01	EF	2	1.87
EF	2	1.34			

Table 3. Candidate 2-itemsets generated after scanning partition P_2 and P_3, respectively.

- Processing of P_3. The weighted count threshold for P_3 is $0.4\times3\times0.536 = 0.643$. All new 2-itemsets generated in P_3 have weighted counts less than 0.643. The itemsets generated in P_1 all have accumulative weighted counts greater than 2.25 and the itemsets generated in P_2, except BE, have accumulative weighted counts greater than 1.45, so they are retained in C_2 and BE is deleted from C_2. The resulting candidate 2-itemsets after processing P_3 are shown in the right part of Table 3.

Therefore, we have

$$\begin{aligned} C_2 &= \{AB, AD, AE, AF, BF, DE, DF, EF\}, \\ C_3 &= \{ABF, ADE, ADF, AEF, DEF\}, \\ C_4 &= \{ADEF\}, \\ C_k &= \emptyset, k \geq 5. \end{aligned}$$

By scanning D once, we have the sets of frequent k-itemsets, L_k, $k \geq 2$, of D along with their counts to be

$$\begin{aligned} L_2 &= \{(AD, 3.61), (AE, 3.21), (AF, 3.61), \\ &\quad (BF, 3.21), (DE, 2.54), (DF, 3.61), \\ &\quad (EF, 2.54)\}, \\ L_3 &= \{(ADE, 2.54), (ADF, 2.94), (AEF, 2.54)\}, \\ L_k &= \emptyset, k \geq 4, \end{aligned}$$

and $L = L_2 \cup L_3$. The association rules can then be obtained from the frequent itemsets in L.

4. Experimental Results

In this section, experimental results obtained with a PC with AMD Athlon XP CPU and 1.0G memory are presented.

4.1. Experiment 1

In this experiment, we compare our fuzzy calendar algebra with the calendar algebra proposed in [16]. The

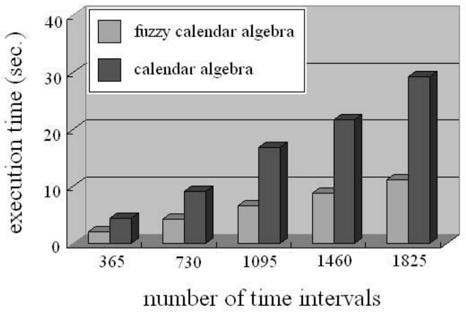

Figure 4. Performance comparison between the calendar algebra and our fuzzy calendar algebra with a simple calendar.

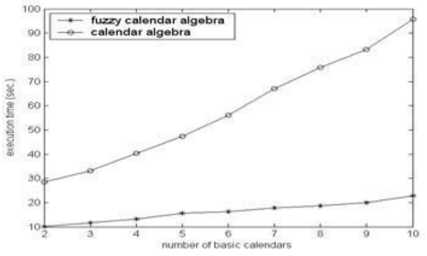

Figure 5. Performance comparison between the calendar algebra and our fuzzy calendar algebra with complicated calendars.

calendar algebra defines operators such as dicing, slicing, and flatten to construct complex calendars. The time intervals are obtained from a specified calendar and the calendric association rules are discovered in these time intervals. For a complex calendar, the calendar algebra decomposes the calendar into basic ones, and then the time intervals of each basic calendar are separately searched. This approach is not efficient since one time interval may be searched with several different basic calendars. On the contrary, our fuzzy calendar algebra checks each time interval only once and can easily find out all the time intervals in a given calendar. Repetitive and unnecessary searches in a time interval can be avoided, and the performance is improved.

We use 1825 time intervals collected in 5 years, with each time interval representing one day in a year. The execution time for finding out the desired time intervals by the calendar algebra and the fuzzy calendar algebra, respectively, is shown in Figure 4. The calendar algebra searches each time interval twice while the fuzzy calendar algebra searches each time interval only once, and thus we can clearly see that our method is more efficient in interpreting calendars. For more complicated calendars, i.e., containing more basic calendars, our fuzzy calendar algebra performs even better, as shown in Figure 5.

4.2. Experiment 2

In this experiment, we'd like to show the effectiveness of our strategy for mining fuzzy temporal patterns. The number of candidates and frequent itemsets discovered are reported. The execution time for discovering the itemsets is also presented. The dataset, $T10.I4.D100K.C10\%$, containing synthetic data [1] is used, where T is the mean size of a transaction, I is the mean size of potential maximal frequent itemsets, D is the number of transactions in units of K, i.e., 1000, and C is the correlation between items in terms of percentage. The dataset is partitioned into 200 partitions and there are 500 transactions in each partition. To infer from these partitions with our fuzzy calendar algebra, we assume the first partition being the time interval of 2003/01/01, the second partition being of 2003/01/02, ..., and so on. The calendar we are interested in is the one given in Eq.(6). The number of candidates and frequent itemsets discovered are given in Table 4 and the execution time for discovering these itemsets is shown in Figure 6. From Table 4 and Figure 6, proper numbers of frequent itemsets can be discovered in acceptable time with our method.

support threshold	number of candidate itemsets	number of frequent itemsets
0.02	1889	1167
0.03	1043	796
0.04	911	633
0.05	635	512
0.06	327	243
0.07	245	182
0.08	179	115

Table 4. The Number of candidates and frequent itemsets discovered.

5. Conclusion

We have presented the fuzzy calendar algebra which supports fuzzy queries on calendars. Fuzzy calendars

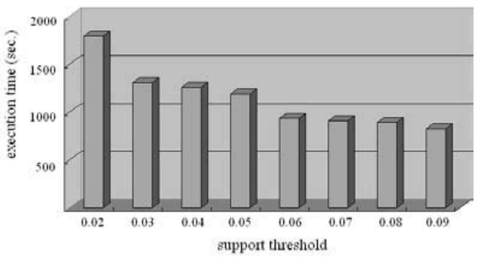

Figure 6. Execution time with different support thresholds.

are natural and easy for use in the sense that they capture the way in which human beings reason with time requirements. To accommodate temporal expressions with multiple time granularities, five operations, **and**, **or**, **not**, **xor**, and **sub**, are introduced.

We have shown the usefulness of the fuzzy calendar algebra by incorporating it with PWM to mine fuzzy temporal association rules from temporal databases. Firstly, the transactions of the database are divided into a sequence of partitions. Then the weight of each partition is inferred, denoting the matching degree of the corresponding time period to the fuzzy calendar specified in the user's queries. Temporal patterns in the divided time periods of different matching degrees are then efficiently produced, and the number of database scans can be effectively reduced. Experimental results have shown that this approach of finding fuzzy temporal association rules is very effective.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the International Very Large Database Conference* , pages 487-499, 1994.

[2] C. Antunes and A. Oliveira. Temporal data mining: an overview. In *Proceedings of KDD Workshop on Temporal Data Mining*, pages 1–13, 2001.

[3] W. H. Au and K. C. C. Chan. Farm: A data mining system for discovering fuzzy association rules. In *Proceedings of the 8th IEEE International Conference on Fuzzy Systems*, pages 1217–1222, 1999.

[4] C. Bettini, C. E. Dyreson, W. S. Evans, and R. T. Snodgrass. A glossary of time granularity concepts. *Temporal Databases: Research and Practice, Lecture Notes in Computer Science*, (1399):406–413, 1998.

[5] C. Bettini and R. D. Sibi. Symbolic representation of user-defined time granularities. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):53–92, 2000.

[6] C. Bettini, X. S. Wang, and S. Jajodia. Temporal semantic assumptions and their use in databases. *IEEE Trans. Knowledge and Data Engineering*, 10(2):277–296, 1998.

[7] R. Chandra, A. Segev, and M. Stonebraker. Implementing calendars and temporal rules in next generation databases. In *Proceedings of 10th International Conference on Data Engineering*, pages 264–273, 1994.

[8] C. E. Dyreson, W. S. Evans, H. Lin, and R. T. Snodgrass. Efficiently supporting temporal granularities. *IEEE Trans. Knowledge and Data Engineering*, 12(4):568–587, 2000.

[9] G. Becher, F. Clerin-Debart and P. Enjalbert. A model for time granularity in natural language. In *Proceedings of the 5th International Workshop on Temporal Representation and Reasoning*, pages 29–36, 1998.

[10] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. *H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining*, chapter Mining Frequent Patterns in Data Streams at Multiple Time Granularities. 2003.

[11] I. A. Goralwalla, Y. Leontiev, M. T. Ozsu, D. Szafron, and C. Combi. Temporal granularity: Completing the puzzle. *Journal of Intelligent Information Systems*, 16(1):41–63, 2001.

[12] C. M. Kuok, A. W. C. Fu, and M. H. Wong. Mining fuzzy association rules in databases. *SIGMOD Record*, 27(1):41–46, 1998.

[13] C. H. Lee, J. C. Ou, , and M. S. Chen. Progressive weighted miner: An efficient method for time-constraint mining. In *Proceedings of 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 449–460, 2003.

[14] I. Merlo, E. Bertino, E. Ferrari, S. Gadia, and G. Guerrini. Querying multiple temporal granularity data. In *Proceedings of the Seventh International Workshop on Temporal Representation and Reasoning*, pages 103–114, 2000.

[15] P. Ning, X. S. Wang, and S. Jajodia. An algebraic representation of calendars. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):5–38, 2002.

[16] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In *Proceedings of the International Very Large Database Conference*, pages 368–379, 1998.

[17] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Trans. Knowledge and Data Engineering*, 14(4):750–767, 2002.

[18] M. D. Soo, R. T. Snodgrass, C. E. Dyreson, C. S. Jensen, and N. Kline. Architectural extensions to support multiple calendars. Technical Report TR-32, Computer Science Department, University of Arizona, 1992.

[19] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[20] R. J. Zhang and E.Unger. Calendar algebra. Technical Report TR-CS-96-1, Kansas State University, 1996.