PERFORMANCE IMPROVEMENTS AND EFFICIENT APPROACH FOR MINING PERIODIC SEQUENTIAL ACCESS PATTERNS

D. Vasumathi
Associate Professor, Department of Computer Science and Engineering,
JNTU College of Engineering.JNTU,
Kukatpally, Hyderabad-500 085, *Vasukumar_devara@yahoo.co.in*
Andhra Pradesh, India.

Dr. A. Govardhan
Principal, JNTU College of Engineering
JNTU,Jagityala, Karimnagar(Dist),
Andhra Pradesh, India

K.Venkateswara Rao
Associate Professor, Department of Computer Science and Engineering,
JNTU College of Engineering.JNTU,
Kukatpally, Hyderabad-500 085,
Andhra Pradesh, India.

Abstract

Surfing the Web has become an important daily activity for many users. Discovering and understanding web users' surfing behavior are essential for the development of successful web monitoring and recommendation systems. To capture users' web access behavior, one promising approach is web usage mining which discovers interesting and frequent user access patterns from web usage logs. Web usage mining discovers interesting and frequent user access patterns from web logs. Most of the previous works have focused on mining common sequential access patterns of web access events that occurred within the entire duration of all web access transactions. However, many useful sequential access patterns occur frequently only during a particular periodic time interval due to user browsing behaviors and habits. It is therefore important to mine periodic sequential access patterns with periodic time constraints. In this paper, we propose an efficient approach, known as TCSMA (Temporal Conditional Sequence Mining Algorithm), for mining periodic sequential access patterns based on calendar-based periodic time constraint. The calendar-based periodic time constraints are used for describing real-life periodic time concepts such as the morning of every weekend. The mined periodic sequential access patterns can be used for temporal-based personalized web recommendations. The performance of the proposed TCSMA is evaluated and compared with a modified version of Web Access Pattern Mine for mining periodic sequential access patterns.

Keywords: Periodic Sequential Access Patterns, Web Access Patterns, Association Rule, Web Log Mining, TCSM&WAPM Algorithm

1. INTRODUCTION

With the explosive growth of resources available on the Internet, web surfing for useful information has become an important daily activity for many users. Web users surf for different web resources according to their needs, knowledge and interests. The discovery and understanding of web users' surfing habits are essential for the development of successful web monitoring and recommendation systems.

Web usage mining aims to discover interesting and frequent user access patterns from web usage data. The discovered knowledge is very useful for modeling users' web access behavior. Statistical techniques have traditionally been used for extracting statistical information from web logs. Association rule mining and sequential pattern mining have also been used to mine association and sequential patterns from web logs for web user access behavior. We can also visualize association and sequential patterns using WUM. However, most of these techniques do not consider the temporal aspect of access patterns.

Web usage mining[1], also known as web log mining, aims to discover interesting and frequent user access patterns from web browsing data that are stored in web server logs, proxy logs or browser logs. The discovered knowledge can be used for many practical applications such as web recommendations, adaptive web sites, and personalized web search and surfing. Many approaches[2-5] have been proposed for discovering sequential patterns from transaction databases. However, most of the pervious works only focused on mining common sequential access patterns of web access events, which occurred frequently within the entire duration of all web access transactions. In practice, many useful sequential access patterns occur frequently only in particular periodic time interval such as the morning of every weekend, but not in other time intervals due to user browsing behaviors and habits. Such sequential access patterns are referred to a periodic sequential access patterns, where periodic time intervals are real-life time concepts such a year, monthly, week and day. With periodic sequential access patterns, we can recommend or predict the occurrence of a web page during a particular time interval.

Recently, temporal association rule mining algorithms[6-8] have been proposed for mining temporal web access patterns. These works have discussed different ways for defining time constraints. However, such algorithms are mainly based on association rules that ignore the sequential characteristics of web access patterns. In addition, these algorithms also encounter the same problem as most Apriori-based algorithms that require expensive scans of database in order to determine which of the candidates are actually frequent. Different from temporal association rule mining, we propose an efficient approach, known as TCSMA (Temporal Conditional Sequence Mining Algorithm), to mine periodic sequential access patterns from web access transaction databases. We also define calendar-based periodic time constraints, which can be used for describing real-life time concept[9].

The rest of this paper is organized as follows. In Section2, we discuss calendar based periodic time constraints. The proposed TECMA is presented in Section3. the experimental results are shown in Section4. finally, the conclusions are given in Section5.

2 CALENDAR-BASED PERIODIC TIME CONSTRAINTS

In this section, we define calendar-based periodic time constraints, which can be used for describing real-life time concepts. The calendar-based periodic time constraints consist of calendar template and calendar instance.

Definition 2.1

A calendar template is defined as $CB_T = (CU_1\ I_1, CU_2\ I_2, \ldots, CU_n\ I_n)$.
Each CU_i is a calendar unit such as year, month, week, day, etc. and I_i is a closed interval that contains all valid time values (positive integers) of CU_i. A calendar template represents a hierarchy of calendar units and valid time intervals. For example, a typical calendar template can be in the form of (year [2007, 2008], month[1, 12], day [1, 31]) or (day-of-week [1, 7], hour [0, 3]).

Definition 2.2

Given a calendar template $CB_T = (CU_1\ I_1, CU_2\ I_2, \ldots, CU_n\ I_n)$, a calendar instance is denoted as $(I_{1'}, I_2', \ldots, I_n')$, where I_I' is an nonempty set of positive integers and $I_I' \subset I_i$, or is a wild-card symbol * that represents all valid time values in I_i. Calendar instances are formed from calendar template by setting some calendar units to specific values. It can then be used for describing real-life time concepts. For **example**,

Given CB_T = (day-of-week [1, 7], hour [0, 23]), we can have
C_I = ({6, 7}, {5, 6, 7, 8}) for the early morning of every weekend or
C_I = (*, {19, 20, 21}) for the evening of everyday.

In practice, some real-life time concepts such as morning or evening may have different meanings to different people depending on their personal behaviors and habits. For example, some people consider that morning is from sunrise to noon, while others consider that it is from 5 AM to 9 AM. Therefore, calendar instances can be defined according to actual practical requirements. We list some special calendar instances based on CB_T = (day-of-week [1, 7], hour [0, 23]) in Table 1.

Time Concept	Calendar Instances
Early morning	(*,{5.6.7.8})
Morning	(*,{9,10,11})
Noon	(*,{12})
Afternoon	(*,{13,14………17})
Evening	(*,{18,19,20,21})
Night	(*,{22,23,0…….4})
Weekdays	({1,2,………5}), *)
Weekend	({6,7}), *)

TABLE 1: Some special calendar instances.

Definition 2.3

A calendar-based periodic time constraint denoted as (C) = $[CB_T, C_I]$ Where CB_T = calendar based template and C_I = one calendar instance For **example**, C = [(day-of-week [1, 7], hour [0, 23]), ({6, 7}, {8, 9})] represents "8:00 AM to 9:59 AM of every weekend".

Given C = [CB_T, C_I], we say time t is covered by C If t belongs to the time interval defined by C. For **example**, t_1 = “2007-11-08 08: 22:45 Saturday” and t_2 = “2007-11-02 09:45:30 Sunday” are covered by C. If we denote the calendar-based periodic time constraint Call = [CB_T, (*, ..., *)], Where CB_T is any calendar based template, then it will specify all the time intervals.

3 THE TCSMA (TEMPORALCONDITIONALSEQUENCE MINING ALGORITHM)

In this section, we discuss our proposed approach, known as TCSMA (Temporal Conditional Sequence mining algorithm), for mining common and periodic sequential access patterns from a given web access transaction database.

3.1 Problem Statement

Generally, web logs can be regarded as a collection of sequences of access events from one user or session in timestamp ascending order. Preprocessing tasks [9] including data cleaning, user identification, session identification and transaction identification can be applied to the original web log files to obtain the web access transactions.

Let UAE = A set of unique access events, (which represents web resources accessed by users, i.e. web pages, URLs, or topics) WAS = A web access sequence
WAS = $e_1e_2\ldots e_n$ ($e_i \in$ UAE for $1 \le i \le n$) is a sequence of access events, and
|WAS| = n is called the length of WAS. Note that it is not necessary that $e_i \neq e_j$ for $I \neq j$ in WAS, that is repeat of items is allowed
WAT = A web access transaction
WAT = (t, WAS), consists of a transaction time t and a web access sequence WAS

All the web access transactions in a database can belong to either a single user (for client-side logs) or multiple users (for server-side and proxy logs). The proposed algorithm does not depend on the type of web logs that contains the web access transactions. Suppose we have a set of web access transactions with the access event set, UAE = {a, b, c, d, e, f}. A sample web access transaction database is given in Table 2.

Transaction Time	Web Access Sequence
2007-11-03 20:21 : 10 Saturday	abdac
2007-11-04 21:45 : 22 Sunday	eaebcac
2007-11-07 18:23 : 24 Wednesday	cacbb
2007-11-10 21:10 : 10 Saturday	babfae
2007-11-10 21:30 : 20 Saturday	afbacfc

TABLE2: A database of web access transactions

WAS = A web access sequence and WAS = $e_1e_2\ldots e_k\ e_{k+1}\ldots e_n$,
WAS $_{prefix}$ = $e_1e_2\ldots e_k$ is called a prefix sequence of WAS, or a prefix sequence of e_{k+1} in WAS.And WAS $_{suffix}$ = $e_{k+1}e_{k+2}\ldots$en is called a suffix sequence of WAS or a suffix sequence of e_k in WAS.
Now A web access sequence (WAS) = WAS $_{prefix}$ + WAS $_{suffix.}$
For **example**,
WAS = abdac can be denoted as WAS = a+bdac = ab+dac = … = abda+c.

Let S_1 and S_2 be two suffix sequences of e_i in WAS, and S_1 is also the suffix sequence of e_i in S_2. Then S_1 is called the sub-suffix sequence of S_2 and S_2 is the super-suffix sequence of S_1. The suffix sequence of e_i in WAS without any super-suffix sequence is called the long suffix

sequence of e_i in WAS. For **example**, if WAS = abdacb, then S_1 = cb is the sub-suffix sequence of S_2 = bdacb and S_2 is the super-suffix sequence of S_1. S_2 is also the long suffix sequence of a in WAS.
Given
WAT_{DB}= A web access transaction database
$WAT_{DB} = \{(t1, S_1), (t2, S_2), \ldots, (t_m, S_m)\}$ in which WAS_i ($1 \le i \le m$) is a web access sequence, and t_i is a transaction time.
Given a calendar-based periodic time constraint C that is defined in Section 2.
$WAT_{DB}(C) = \{(t_i, WAS_i) \mid t_i$ is covered by C, $1 \le l \le m\}$ is a subset of WAT_{DB} under C.
$|WAT_{DB}(C)|$ is called the length of WAT_{DB} under C. The support of WAS in WAT_{DB} under C is defined in equation (3.1).

$$Sup(WAS,C)= \frac{|\{S_i \mid WAS \in S_i,(t_i,S_i) \in WAT_{DB}(C)\}|}{|WAT_{DB}(C)|} \qquad 3.1$$

A web access sequence WAS is called a periodic sequential access pattern,
if sup(WAS, C) ≥MinSup, where MinSup is a given support threshold.
Let's consider the sample database in Table 2.
Suppose MinSup = 75% and calendar-based periodic time constraint
C =[(day-of-week [1, 7], hour [0, 23]), ({6, 7}, {20, 21})].
It is required to find all web access patterns supported by at least 75% access sequences within the time interval from 8:00 PM to 9:59 PM of every weekend from the sample database. If we use Call as the calendar-based periodic time constraint, the mining results should be all common sequential access patterns satisfying the given support threshold.

3.2 Proposed Approach

As shown in Figure:1, the proposed TCSMA consists of the following steps:
(1) Constraint Preprocessing;
(2) Constructing Event Queues for Conditional Sequence Base;
(3) Single Sequence Testing for Conditional Sequence Base;
(4) Constructing Sub-Conditional Sequence Base; and
(5) Recursive Mining for Sub-Conditional Sequence Base.

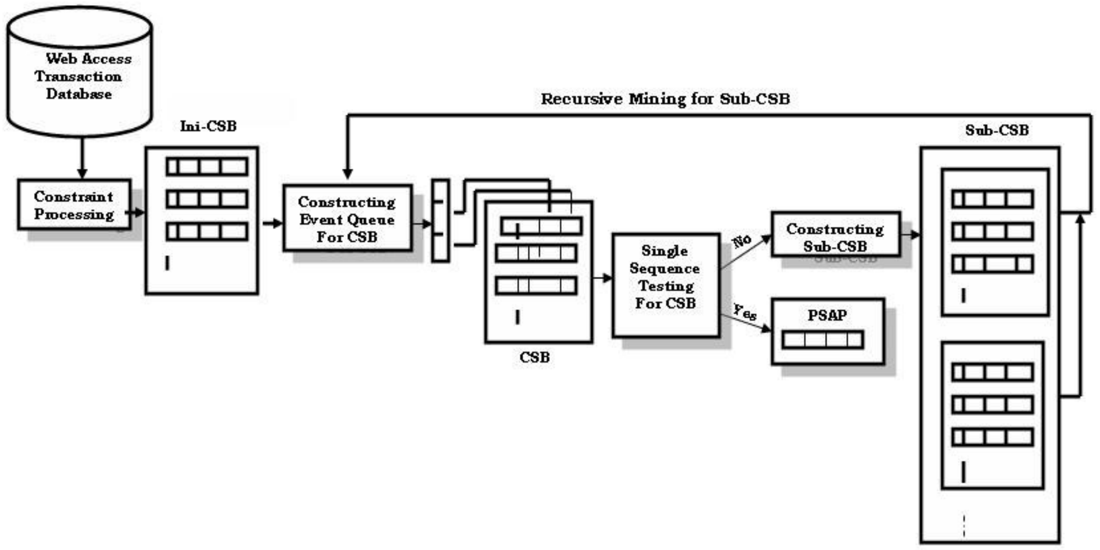

Web Access Transaction Database
Constraint Processing
Ini-CSB
Constructing Event Queue For CSB
Recursive Mining for Sub-CSB
CSB
Single Sequence Testing For CSB
No
Yes
Constructing Sub-CSB
PSAP
Sub-CSB

FIGURE 1: Overview of the proposed TCSMA

3.2.1 Constraint Preprocessing

The first step in the TCSMA is to filter the web access transaction database by discarding all transactions that do not satisfy the given calendar-based periodic time constraint. The remaining constraint-satisfied (S_c) transactions are then used to construct the initial conditional sequence base. The initial conditional sequence base and conditional sequence base are defined as follows.

Definition 3.1

The initial conditional sequence base, denoted as Ini-CSB, is the set of all constraint-satisfied transactions in the given web access transaction database, where constraint-satisfied transactions are web access transactions whose transaction times are covered by the given calendar-based periodic time constraint.

Definition 3.2

The conditional sequence base of an event e_i based on prefix sequence WAS_{prefix}, denoted as CSB(Sc), where

Sc = $WAS_{prefix} + e_i$, is the set of all long suffix sequences of e_i in sequences of a certain dataset.

If $WAS_{prefix} = Ø$, the dataset is equal to the initial conditional sequence base of the given web access transaction database. Otherwise, it is the conditional sequence base CSB(WAS_{prefix}).

We also call CSB (WAS c) the conditional sequence base of conditional prefix Sc. The initial conditional sequence base can also be denoted as CSB(Ø), with Sc =Ø. The ConsPreprocessing algorithm for constraint preprocessing of transactions from the web access transaction database WAT_{DB} is given in Figure: 2.

```
Algorithm: Cons Preprocessing

Input:
1: C = [CB_T, C_I] – calendar-based periodic time constraint that consists of calendar based
template CB_T and calendar instance C_I
2: WAT_DB = {WAT_i |WAT_i = (t_i, WAS_i), 1 ≤ i ≤ n} – web access transaction database, and WAT_i is
a web access transaction that consists of transaction time t_i and web access sequence WAS_i
Output:
1: Ini-CSB - initial conditional sequence base of WAT_DB
Method:
1: Initialize Ini-CSB = Ø.
2: For each WAT_i ∈ WAT_DB, if t_i is covered by C, insert WAS_i into Ini-CSB.
3: Return Ini-CSB.
```

FIGURE 2: The algorithm for constraint preprocessing of transactions.

Example: Given a calendar-based periodic time constraint
C = [(day-of-week [1, 7], hour [0, 23]), ({6, 7}, {20, 21})], as the time of the third transaction in Table 3.2 is “2007-11-05 18:23:24 Wednesday”, it is not covered by C. So the web access sequence bbcac is discarded. After preprocessing, the Ini-CSB of the sample database contains {abdac, eaebcac, babfae, afbacfc}.

3.2.2 Constructing Event Queues for Conditional Sequence Base

The second step of the TCSMA is to construct event queues for CSB(Sc) (for Ini-CSB, Sc = Ø). The process performs the following four steps:

(1) Finding conditional frequent events from CSB(Sc);
(2) Creating a Header Table;
(3) Constructing event queues; and
(4) Deleting non-frequent events.

The conditional frequent event is defined as follows.
Definition 3.3

The conditional frequent event is the event whose support in the given conditional sequence base is not less than the support threshold, MinSup. To find conditional frequent events in CSB(Sc), we need to identify those events with support of greater than or equal to MinSup. This is given in equation (3.2) below.

$$Sup(e_i)= \frac{|\{S_j|\ e_j \in S_j, S_j \in CSB(S_c)\}|}{|\ \text{Ini-CSB}\ |} \geq \text{MinSup} \qquad 3.2$$

In equation (3.2), |{Sj | ei ∈ Sj, Sj ∈ CSB(Sc)}| is the number of sequences which contains the item labeled ei in CSB(Sc), and |Ini-CSB| is the length of Ini-CSB. Then, all the conditional frequent events form the entire Header Table of CSB(Sc). A linked-list structure for each conditional frequent event e_i, called e_i–queue, is created. Each item of e_i–queue is the first item labeled e_i in sequences of CSB(Sc). The head pointer of each event queue is recorded in the

Header Table. Finally, as all the items of sequences in CSB(Sc) which are labeled as non-frequent events are not needed anymore, they are discarded. The ConstructEQ algorithm for constructing event queues for CSB(Sc) is given in Figure:3

```
Algorithm: Construct EQ

Input:
1: MinSup - support threshold
2: CSB(Sc) - conditional sequence base of Sc
3: UAE = {ei|1 ≤ i ≤ n} – all access events in CSB(Sc)
Output:
1: CSB(Sc) with Header Table HT and event queues
Method:
1: Create an empty Header Table HT for CSB(Sc).
2: For each $e_i \in$ UAE, if sup($e_i$) ≥MinSup, insertn $e_i$ into HT.
3: For each conditional sequence $\in$ CSB(Sc) do
a) For each $e_i \in$ HT, insert the first item labeled ei in this sequence into $e_i$ -queue.
b) Delete all items of events $\notin$ HT from this sequence.
4: Return CSB(Sc) with HT and event queues.
```

FIGURE 3: The algorithm for constructing event queues for CSB.

Example For the Ini-CSB = {abdac, eaebcac, babfae, afbacfc}, the results after constructing the Header Table and event queues is given in Figure:4 Each access event is denoted as (event: count), where event is the event name and count is the number of sequences which contains the item labeled as event in Ini-CSB. To be qualified as a conditional frequent event (with MinSup = 75% and |Ini-CSB| = 4), an event must have a count of at least 3. Therefore, the conditional frequent events are (a:4), (b:4) and (c:3). The a-queue, b-queue and c-queue are shown by the dashed lines starting from the Header Table. The items labeled as non-frequent events d, e and f in each sequence are deleted. Similarly, for any subsequent conditional sequence base, the Header Table and event queues can also be constructed using the ConstructEQ algorithm.

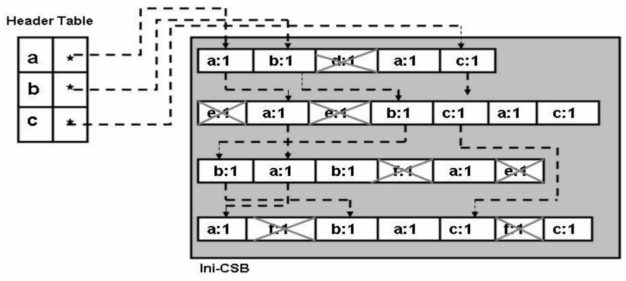


FIGURE 4: Ini-CSB with the Header Table and event queues.

3.2.3 Constructing Sub-Conditional Sequence Base

The sub-conditional sequence base is defined as follows.

Definition 3.4

CSB(WAS $_{prefix}$ +e $_i$) is called the sub-conditional sequence base of CSB(WAS $_{prefix}$), if $e_i \neq Ø$ for each access event e_i in the Header Table of CSB(Sc), the ConstructSubCSB algorithm for constructing CSB(Sc+e_i) based on CSB(Sc) is given in Figure:5

Algorithm: ConstructSubCSB

```
Input:
1: CSB(Sc) - conditional sequence base of Sc
2: $e_i$ - a given event in Header Table of CSB(Sc)
Output:
1: CSB(Sc+$e_i$ ) - conditional sequence base of $e_i$ based on CSB(Sc)
Method:
1: Initialize CSB(Sc+$e_i$ ) = Ø.
2: For each item in ei-queue of CSB(Sc), insert its suffix sequence into CSB(Sc+$e_i$ ).
3: Return CSB(Sc+$e_i$ ).
```

FIGURE 5: The algorithm for constructing Sub-CSB.

Example For the Ini-CSB shown in Figure:4, we obtain all suffix sequences of a by following the a-queue as CSB(a), which is one of the sub-conditional sequence base of Ini-CSB. The result is shown in Figure:6 CSB(a) contains {bac:1, bcac:1, ba:1, bacc:1}. Note that bac:1 is the abbreviation of (b:1)(a:1)(c:1).

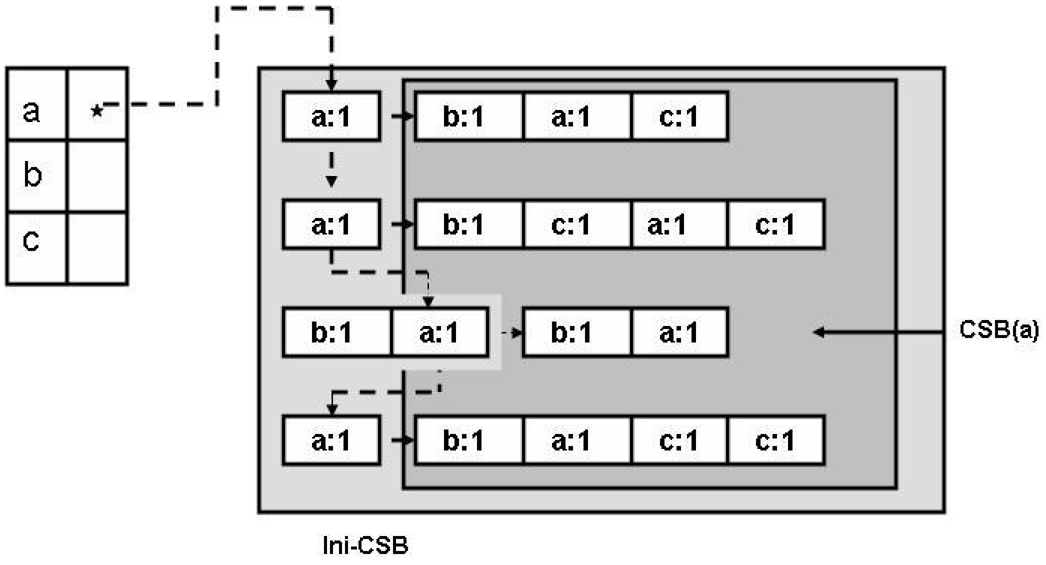


FIGURE 6: Construction of CSB(a) based on Ini-CSB.

3.2.4 Single Sequence Testing for Conditional Sequence Base

In this step, if all the sequences in CSB(Sc) can be combined into a single sequence, the mining of CSB(Sc) will be stopped. This single sequence will be used to form a part of the final periodic sequential access patterns. Otherwise, we construct Sub-CSBs for CSB(Sc) and perform recursive mining. The TestCSB algorithm for testing whether all the sequences in CSB(Sc) can be combined into a single sequence is given in Figure:7

Algorithm: TestCSB

```
Input:
1: CSB(Sc) – conditional sequence base of Sc
2: HT – Header Table of CSB(Sc)
Output:
1: test result - successful or failed flag
2: SingleSeq - single sequence of CSB(Sc)
Method:
1: Initialize SingleSeq = Ø.
2: If CSB(Sc) = Ø, return successful and SingleSeq = Ø.
3: For i = 1 to maximum length of sequences  CSB(Sc) do
   a) If all the ith items in each sequence  CSB(Sc) are the same event e. And if total count of
   these items ≥ MinSup X |Ini-CSB|, create a new item e with the count and insert it into
   SingleSeq.
   b) Otherwise, return failed and SingleSeq = Ø.
4: Return successful and SingleSeq
```

FIGURE 7: The algorithm for testing conditional sequence base.

Example For CSB(a) = {bac:1, bcac:1, ba:1, bacc:1}, the first item of each sequence can be combined into one item (b:4), but the second item cannot. The combination is stopped and returns the failed flag. For CSB(aa) = {c:2, cc:1}, the sequences can be combined into a single sequence c:3 and the successful flag is returned.

3.2.5 TCS-mine for Mining Periodic Sequential Access Patterns

The complete TCSM algorithm is shown in Figure:8

```
Algorithm: TCSM

Input:
1: C = [$CB_T$, $C_I$ ] – calendar-based periodic time constraint that consists of calendar
template $CB_T$ and calendar instance $C_I$
2: MinSup - support threshold
3: $WAT_{DB}$ = {$WAT_i$ |$WAT_i$ = ($t_i$, $WAS_i$ ), 1 ≤ i ≤ n} – web access transaction database,
and $WAT_i$ is a web access transaction that consists of transaction time $t_i$ and web access
sequence $WAS_i$
4: E = {$e_i$|1 ≤ i ≤ n} – all access events in $WAT_{DB}$
Output:
1: PSAP - the set of periodic sequential access patterns
Method:
1: Initialize PSAP = Ø.
2: Use ConsPreprocessing to construct Ini-CSB (CSB(Sc), Sc = Ø).
3: Use ConstructEQ to construct event queues for CSB(Sc).


4: Use TestCSB to test single sequence for CSB(Sc).
   a) If test is successful, insert all ordered combinations of items in
      frequent sequence FS = Sc+SingleSeq into PSAP.
   b) Otherwise, for each $e_j$ in Header Table of CSB(Sc), use ConstructSubCSB to construct
         CSB(Sc+$e_j$ ). Set Sc = Sc+$e_j$ and recursively mine CSB(Sc) from step3.
5: Return PSAP.
```

FIGURE 8: The algorithm for mining periodic sequential access patterns.

Length of Patterns	Periodic Sequential Access Patterns
1	a:4, b:4, c:3
2	aa:4, ab:4, ac:3, ba:4, bc:3
3	aac:3, aba:4, abc:3, bac:3
4	abac:3

TABLE 3: The periodic sequential access patterns of the sample database.

Example The complete periodic sequential access patterns with C = [(day-of-week [1, 7], hour [0, 23]), ({6, 7}, {20, 21})] and MinSup = 75% is shown in Table 3.

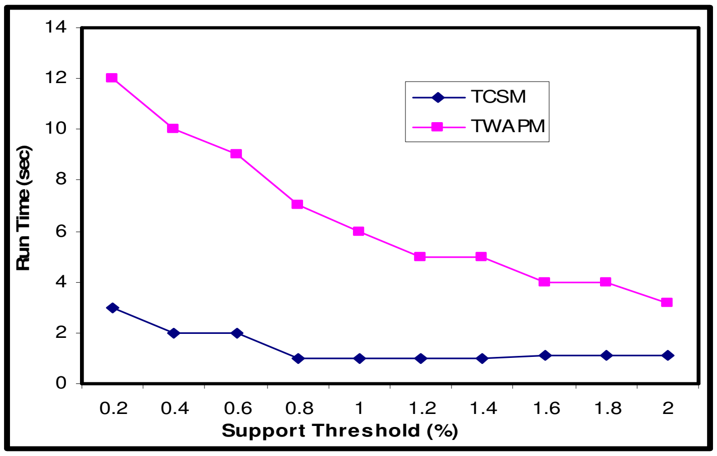
4 PERFORMANCE EVALUATION

In this section, we present the performance of TCSM and compare it with the temporal version of the Web Access Pattern mine (or TWAPM) algorithm for mining periodic sequential access patterns. Web Access Pattern mine is one of the most efficient algorithms that mine common sequential access patterns from a highly compressed data structure known as Web Access Pattern- tree. As evaluated in the performance of the Web Access Pattern mine algorithm is an order of magnitude faster than other Apriori-based algorithms. Therefore, we only compare the TCSM algorithm with the TWAPM algorithm here.

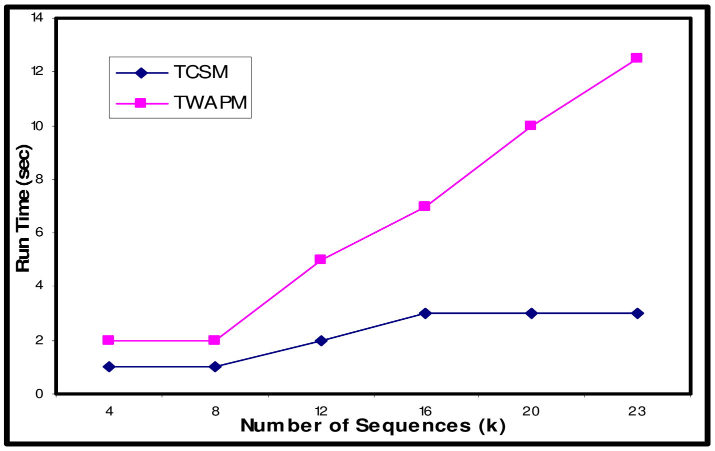
In order to deal with calendar-based periodic time constraints, the step on Constraint Preprocessing discussed in Section 3.2.1 is applied to TWAPM for extracting all the constraint-satisfied transactions from the original web access transaction database. The Web Access Pattern-tree is then constructed from the constraint-satisfied transactions, and the Web Access Pattern mine algorithm is used to mine the periodic sequential access patterns.

The two algorithms, TCSM and TWAPM, are implemented in Java. All experiments are performed on a 3.00 GHz Intel Pentium 4 PC machine with 512 MB memory, running on Microsoft Windows XP Professional. The Microsoft Anonymous Web Data is used to test the two algorithms. This dataset contains logs on which areas of www.microsoft.com each user has visited and has a total of 32,711 transactions, with each transaction containing from 1 up to 35 page references from a total of 294 pages. We set the calendar-based periodic time constraint C = [(day-of-week [1, 7], hour [0, 23]), ({1, 2, …, 5}, *)], which means every hour of every weekday. As a result, 22,717 constraint-satisfied transactions are used for the measurement.

To measure the performance, two experiments have been conducted. In the first experiment, we have measured the scalability of the two algorithms with respect to different support thresholds. This experiment uses the 22,717 constraint-satisfied web access sequences with different support thresholds (from 0.2% to 2.4%). The experimental results in Figure:9 (a) have shown that the run time of the TWAPM increases sharply, when the support threshold decreases, and the TCSM always costs less time than the TWAP-mine. In the second experiment, we have measured the scalability of the two algorithms with respect to different sizes of the constraint-satisfied web access sequences. The experiment uses a fixed support threshold (0.2%) with different databases (with sizes vary from 4,000 to 22,717 constraint-satisfied web access sequences). The experimental results in Figure:9 (b) have shown that the TCSM has better scalability than the TWAPM while the size of input database becomes larger.



(a)



(b)

FIGURE 9(a&b): Scalability with different (a) support thresholds (b) number of sequences.

5. CONCLUSIONS

In this paper, we have proposed an efficient approach, known as TCSMA for mining periodic sequential access patterns based on calendar-based periodic time constraints that can be used for describing real-life time concepts. The performance of the TCSMA has been evaluated and compared with a temporal version of the Web Access Pattern – mine algorithm. Experimental results have shown that the TCSMA performs much more efficient than the TWAPMA,especially when the support threshold becomes small and the number of web access sequences gets larger.

REFERENCES

FOR JOURNALS:

[1] Kosala R., and Blockeel H., (2000). Web Mining Research: A Survey. In *ACM SIGKDD Explorations*, Vol. 2, pp. 1-15.

[2] Ganter B., and Wille R., (1999). Formal Concept Analysis: Mathematical Foundations. Springer, Heidelberg, 1999.

[3] Cooley R., Mobasher B., and Srivastava J. (1999). Data Preparation for Mining World Wide Web Browsing Patterns. In *Journal of Knowledge and Information Systems*, Vol. 1, No. 1.

FOR CONFERENCES:

[4] Agrawal R., and Srikant R. (1995). Mining Sequential Patterns. In *Proceedings of the 11th International Conference on Data Engineering*, Taipei, Taiwan, pp. 3-14.

[5] Srikant R., and Agrawal R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT)*, Avignon, France, pp. 3-17.

[6] Pei J., Han J., Mortazavi-asl B., and Zhu H. (2000). Mining Access Patterns Efficiently from Web Logs. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '00)*, Kyoto, Japan, pp. 396-407.

[7] Lu H., Luo Q., Shun Y.K., (2003). Extending a Web Browser with Client-Side Mining. In *Proceedings of the 5th Asia Pacific Web Conference (APWeb)*, pp. 166-177.

[8] Ozden B., Ramaswamy S., and Silberschatz A. (1998). Cyclic Association Rules. In *Proceedings of the 14th International Conference on Data Engineering*, pp. 412-421.

[9] Ramaswamy S., Mahajan S., and Silberschatz A. (1998). On the Discovery of Interesting Patterns in Association Rules. In *Proceedings of the 24th International Conference on. on Very Large Data Bases*, New York, USA, pp. 368-379.