E-MAP: Efficiently Mining Asynchronous Periodic Patterns

Fahad Maqbool, Shariq Bashir, and A. Rauf Baig

FAST-National University of Computer and Emerging Sciences, Islamabad, Pakistan
A.K. Brohi Road, H-11/4, Islamabad, Pakistan

Summary
Mining periodic patterns in temporal dataset plays an important role in data mining and knowledge discovery tasks. In this paper, we propose a novel algorithm E-MAP (Efficient Mining of Asynchronous Periodic Patterns) for efficient mining of asynchronous periodic patterns in large temporal datasets. Our proposed algorithm discovers all maximal complex patterns in a single step and single scan without mining single event and multi events patterns. To check the effectiveness of our approach, we also provide detailed experimental results on real and artificial large temporal datasets. Our experimental results suggest that mining asynchronous periodic patterns using our proposed algorithm is fast and efficient than as compared to previous approach SMCA, which is a three-step based algorithm for mining maximal complex patterns and requires depth-first-enumeration for mining multi events and maximal complex patterns.

Key words:
Asynchronous periodic patterns, temporal data mining, maximal complex patterns, knowledge discovery.

Introduction

Mining periodic patterns in temporal datasets play an important role in data mining and knowledge discovery tasks, and have been successfully applied in many data mining applications. The periodic patterns exist in many kinds of data, for example transactional datasets, daily traffic patterns, meteorological data, stock data, event logs, web logs and power consumptions are all major applications of periodic patterns mining. In last one decade, mining periodic patterns have been studied in various domains such as sequential patterns [5], temporal patterns [1], cyclic association rules [4], surprising patterns [2] and asynchronous periodic patterns [3, 6]. Each of these studies have a different definition for periodicity, which requires different kind of techniques and algorithms.

Yang et al. in [6] introduced the problem of mining asynchronous periodic pattern for mining longest periodic subsequence which may contain a disturbance of length up to a certain threshold. Two parameters, namely *min_rep* and *max_dis* are employed to qualify valid patterns and the symbol subsequence containing it. However their model has several problems such as lack of finding multiple events at one time slot and lack of finding successive non-overlapped segments. To address these problems, in [3] K. Huang proposed a novel SMCA algorithm which requires no candidate pattern generation as compared to previous technique [6]. Their algorithm allows the mining of all asynchronous periodic patterns, not only in a sequence of events, but also in a temporal dataset with multiple event sets. In [3] they also proposed a dynamic hash-based validation mechanism which discovers all asynchronous type periodic patterns in a single scan of temporal dataset. Their four phase approach uses a sequence of algorithms to mine singular pattern, multiple pattern, maximal complex pattern and finally asynchronous periodic patterns. Each of these algorithms uses output of the last executed algorithm as their input. The main limitation of their algorithm is that, it not only mines the maximal complex pattern but also its subsets (single event patterns and multiple events patterns) using depth first search enumeration approach, thus wasting a considerable amount of processing time for mining subsets. For large datasets having *i-patterns* where *i* is too large, a large amount of processing time is waste for mining singular and multi events *1-patterns* that are subsets of *i-patterns*. To increase the efficiency of mining asynchronous periodic patterns on large datasets, we propose a novel efficient algorithm E-MAP. Our propose algorithm finds all maximal complex patterns in a single step algorithm using a single dataset scan without mining single event and multiple events patterns explicitly, while asynchronous periodic patterns are mined using the same depth first search enumeration process as described in [3]. The single dataset scan and single step mining approach makes the E-MAP much faster and efficient as compared to previous technique SMCA. The other feature of E-MAP is that, it requires less storage space as compared to SMCA. To check the effectiveness of our E-MAP approach, we also provide detailed experimental results on real and artificial datasets. Our different experimental results suggest that mining asynchronous periodic patterns using E-MAP is more efficient as compared to SMCA.

2. Related Work

Much work has been done in the field of periodicity detection in time series databases. Most of the work done is related to synchronous pattern mining and less attention has been paid to asynchronous pattern mining. Yang et al. [6] proposed a flexible model of asynchronous periodic patterns to mine patterns that are

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

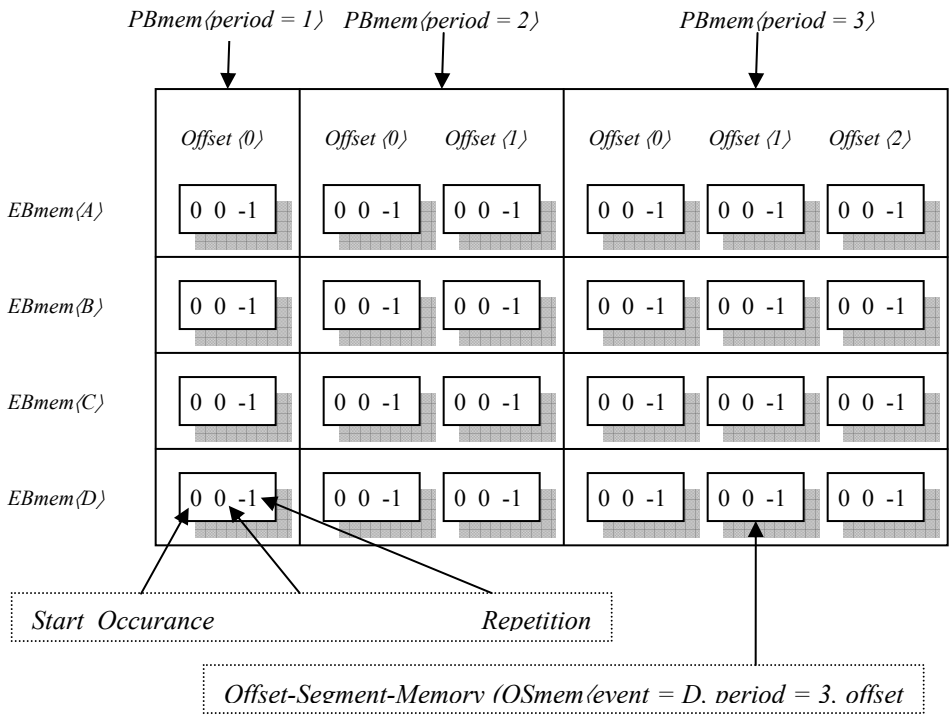


Fig. 1. Graphical Representation of *Event-Block-Memories (EBmem)*, *Period-Block-Memories (PBmem)* and *Offset-Segment-Memories (OSmem)* of Table 1 dataset

of any length and may only be present within a subsequence, and whose occurrences may be shifted due to disturbance. Two parameters *min_rep* and *max_dis* are employed to specify the minimum number of repetitions required within each contiguous segment of pattern occurrences and the maximum disturbance allowed between any two successive valid segments. Upon satisfying these two requirements, the longest valid subsequence of a pattern is returned. A two phase algorithm is devised to first generate potential periods by distance-based pruning followed by an iterative procedure to derive and validate candidate patterns and locate the longest valid subsequence.

K. Huang et al. in [3] proposed a novel asynchronous partial periodic patterns mining algorithm in multi-event temporal database. Three parameters, namely *min_rep (minimum repetitions)*, *global_rep (global repetitions)* and *max_dis (maximum disturbance)* are employed to qualify valid patterns and the subsequence containing them, where this subsequent in turn can be viewed as a list of valid segments of perfect repetitions interleaved by a disturbance. Each valid segment is required to be of at least *min_rep* contiguous repetitions of the pattern and the distance of each piece of disturbance is allowed only up to *max_dis*. The overall number of repetitions of a sequence is equal to the sum of the repetitions of its valid segments. A sequence is termed valid if and only if the overall repetitions of the pattern are greater than *global_rep*. They proposed a four-phase algorithm for mining asynchronous periodic pattern. They first introduce a hash based validation mechanism to discover all single event periodic patterns, named SPMiner (Single event pattern validation). In order to generate the multi-event periodic pattern, complex pattern and asynchronous sequences, they employ depth first enumeration approach to develop MP-Miner (Multiple event pattern validation), CPMiner (Complex pattern validation) and APMiner (Asynchronous pattern validation).

3. Algorithm Overview

Table 1: A temporal data with 4 events ⟨*A, B, C, D*⟩ and 7 time instances

Time instance	Event
1	A,C
2	A,B
3	A,C
4	A,B,D
5	D
6	C
7	C

Many of the previously proposed algorithms [3, 6] require multiple steps to mine complex patterns. These steps are executed in a manner that discover simple patterns (single event patterns and multiple events patterns) in early stages of algorithm and then merge them to form more maximal complex patterns using depth first search enumeration and finally asynchronous periodic patterns in the later stage, thus requiring multiple steps and large

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

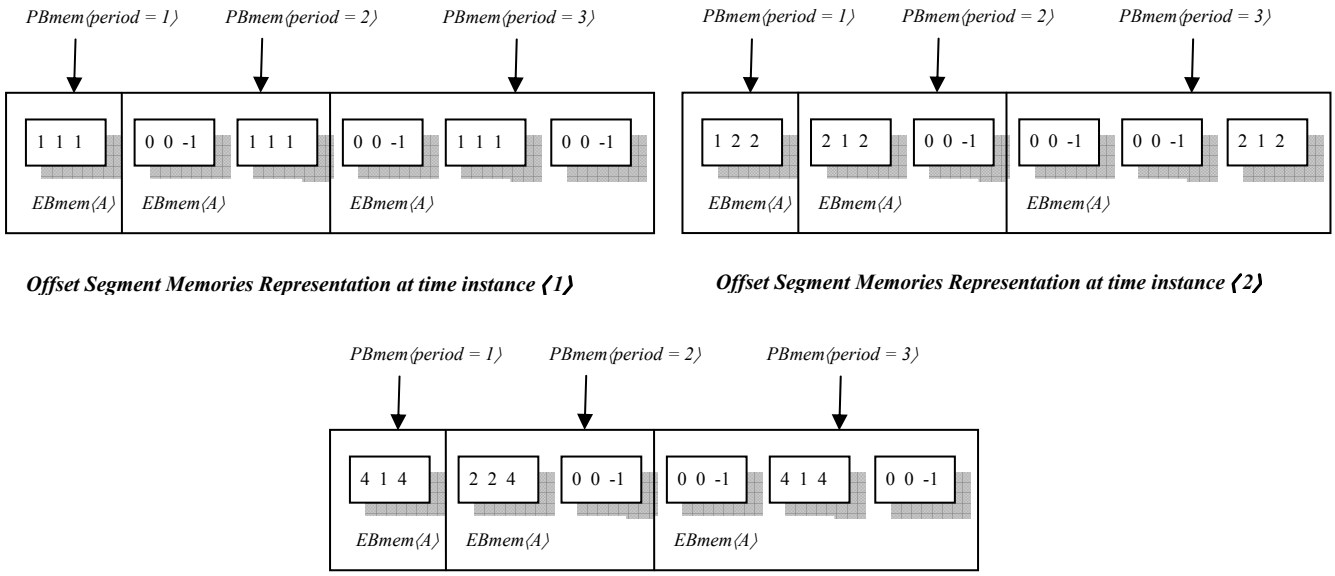


Fig. 2. Graphical Representation of *Offset-Segment-Memories* modification with time instance ⟨*1,2,4*⟩ of event *A* with $L_{max} = 3$.

processing time. Our proposed algorithm E-MAP finds all maximal complex patterns in a single step algorithm without mining single event and multiple events patterns using only one temporal dataset scan. The other main feature of main E-MAP is that, it requires less storage space as compared to SMCA [3]. In SMCA patterns generated after each step of the algorithm requires the storage space for the intermediate output results. In contrast E-MAP is a single step algorithm; therefore it does not require storing any intermediate output results. The storage requirements of the E-MAP algorithm are much less than as compared to the space required by the SMCA algorithm.

The E-MAP algorithm we introduced the following concepts for its data structure representation.

Event-Block-Memory: - In E-MAP each individual event of temporal dataset contains its own *Event-Block Memory* (*EBmem*), which is associated with each periods *P* in L_{max} *(1≤ P ≤* L_{max}*)* and contains internal *Offset-Segment-Memories* with respect to different offsets of *P (0≤ offset < P)*. For example the dataset of Table 1 contains four different events ⟨*A, B, C, D*⟩ so Event-Block-Memory is associated with *EBmem⟨A⟩*, *EBmem⟨B⟩*, *EBmem⟨C⟩* and *EBmem⟨D⟩*. Figure 1 shows the graphical representation of *Event-Block-Memories* of Table 1 dataset.

Period-Block-Memory: - As each individual event contains its own Event-Block-Memory, same as each individual period *P* of L_{max} contains its own *Period-Block-Memory* (*PBmem*), which contains its own internal *Offset-Segment-Memories* for all events of datasets with respect to different offset of *P (0≤ offset < P)*. For example with L_{max} *= 3,* there are *3* different time periods associated with it ⟨*P =1, P=2, P=3*⟩. Therefore *Period-Block-Memory* is associated with *PBmem⟨P=1⟩, PBmem⟨P=2⟩* and *PBmem⟨P=3⟩*. Figure 1 shows the graphical representation of Period-Block-Memories of this example.

Offset-Segments-Memory: - *As* we have explained *EBmem* is association with each period *P* of L_{max} and contains internal *Offset-Segment-Memories* with respect to different offsets of *P*. For a time period of length *P* there are exactly *P (0≤ offset < P) Offset-Segment-Memories* are association with each individual *EBmem*. For example if ⟨*P = 2*⟩ and ⟨*event = B*⟩, then there are exactly *2 Offset-Segment-Memories* are associated with *EBmem⟨B⟩. Offset-Segment-Memory* of *OSmem⟨event = B, period = 2, offset = 0⟩* and *OSmem⟨event = B, period = 2, offset = 1⟩*. Same as for ⟨*P = 3*⟩ and ⟨*event = A*⟩ there are exactly *3 Offset-Segment-Memories* are associated with *EMmem⟨A⟩. Offset-Segment-Memory* of *OSmem⟨event= A, period = 3, offset = 0⟩*, *OSmem⟨event = A, period = 3, offset = 1⟩* and *OSmem⟨event = A, period = 3, offset = 2⟩*. Figure 1 shows the graphical representation of *Offset-Segment-Memories* of Table 1 dataset events with $L_{max} = 3$.

Offset-Segment-Memory Components: - Each individual *Offset-Segment-Memory* contains three main components *Start_Occurance*, *Repetition* and *Last_Occurance* that are used in mining maximal complex periodic patterns. At start of the E-MAP algorithm all *Start_Occurance* and *Repetition* components of *Offset-Segment-Memories* are initialize to *0* and *Last_Occurance* are set to *-1*, then after the following rule is applied for their modifications during pattern mining.

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

Rule 1:- If *Last_Occurance* of an *Offset-Segment-Memory* contains a difference with respect to current time instance, which is equal to its *PBmem* length, then *Last_Occurance* is set to current time instance and *Repetition* is increment by *1* and *Start-Occurance* is left unchanged. If *Last_Occurance* of an *Offset-Segment-Memory* contains a difference with respect to current time instance, which is greater than its *PBmem* length, then *Last_Occurance* and *Start_Occurance* are set to current time instance and *Repetition* is set to *1*.

Example 1:- Let we take an example with $L_{max} = 3$, *event = A* and time instances *⟨1, 2, 4⟩*. At time instance *⟨1⟩* first we calculate the offset difference of current time instance with respect to all periods in L_{max}. For *⟨P = 1⟩* the offset will be *0 ⟨offset = current time instance % P⟩,* for *⟨P = 2⟩* the offset will be *1 ⟨offset = current time instance % P⟩* and for *⟨P = 3⟩* the offset will be *1*. For *⟨P = 1⟩* and time instance *⟨1⟩* since the offset is equal to *0, therefore,* the components of *Offset-Segment-Memory* of *OSmem⟨ event = A, P = 1, offset = 0⟩* will be modify. As the difference between *Last_Occurance* and current time instance is equal to *PBmem⟨P = 1⟩* length *(current time instance - Last_Occurance = 1 – (-1) = 1)*, hence according to Rule 1 *Last_Occurance* and *Start_Occurance* are set to current time instance *⟨Last_Occurance = 1, Start_Occurance = 1⟩* and *Repetition* is increment by *1 ⟨Repetition = 1⟩*. Similarly for *⟨P = 2⟩* and time instance *⟨1⟩* the *offset = 1,* so the components of *Offset-Segment-Memory* of *OSmem⟨ event = A, P = 2, offset = 1⟩* will be modify. Since the difference between *Last_Occurance* and current time instance is greater to *PBmem⟨P = 1⟩* length, therefore again according to Rule 1 *Last_Occurance* and *Start_Occurance* are set to current time instance *⟨time instance = 1⟩* and *Repetition* is increment by *1⟨Repetition = 1⟩*. The same above process will be repeat for *⟨P = 3⟩ offset = 1* and time instance *⟨1⟩*.

Let we execute our example for time instance *⟨2⟩*, *⟨P = 1⟩* with *offset = 0*. As the *offset = 0* so the components of *Offset-Segment-Memory* of *OSmem⟨ event = A, P = 1, offset = 0⟩* will be modify. Since the difference between *Last_Occurance* and current time instance is equal to *PBmem⟨P = 1⟩* length, therefore according to
Rule 1 *Repetition* is increment by *1* and set to *⟨Repetition = 2⟩* and *Last_Occurance* is set to current time instance *⟨Last_Occurance = 2⟩*. Rest of the example will follow the same process; Figure 2 shows the complete graphical representation of *Offset-Segment-Memories* components modification at each time instance with event *⟨A⟩*.

4. (E-MAP): Efficiently Mining of Asynchronous Periodic Patterns

```
E-MAP ( )

1.  for ∀ e ∈ E
2.     for ∀ p ∈ P
3.     OSmem⟨event=e, P=p, offset=i⟩. Start_Occurance = 0
4.     OSmem⟨event=e, P=p, offset=i⟩. Last_Occurance = -1
5.     OSmem⟨event=e, P=p, offset=i⟩. Repetition = 0

6.  for ∀ t ∈ D of temporal dataset
7.     for ∀ e ∈ E that occurs at time instant t
8.      for ∀ p ∈ Lmax
9.      Calculate oset=t % p
10. if ((t - OSmem⟨event=e, P=p, offset=oset⟩. Last_Occurance)>
        p)then
11.     {
12.     AddInOffset-SegmentModifyList(OSmem⟨event = e, P = p,
        offset = oset));
13.     OSmem⟨event=e, P=p, offset=oset⟩. Last_Occurance = t
14.     OSmem⟨event=e, P=p, offset=oset⟩. Start_Occurance = t
15.     OSmem⟨event=e, P=p, offset=oset⟩..Repitition=1
16      }
17.    else
18.    {
19.     OSmem⟨event=e, P=p, offset=oset⟩. Last_Occurance=t
20.     OSmem⟨event=e, P=p, offset=oset⟩. Repitition++
21.     }



22. for each Offset-Segment S in AddInOffset-SegmentModifyList
23.       Combine S in valid sequence if it is non-overlap segment
24.           if ( valid sequence contains a Repetitions greater than
              or equal to global_rep and contain a disturbance less
              than max_dis ) then
25.               Add in MaximalComplexPattern list
26.    for ∀ p ∈ Lmax
27.       for ∀ e ∈ E
28.          for each offset oset ∈ p
29.             S = OSmem⟨event =e P = p offset = oset⟩
30.             If S.Repetitions > min_rep Then
31.                Combine S in valid sequence if it is non-overlap
                   segment
32.                if ( valid sequence contains a Repetitions greater
                        than or equal to global_rep and contain a
                        disturbance less than max_dis ) then
33.                        Add in MaximalComplexPattern list
```

Fig. 3. Pseudo code of E-MAP Algorithm.

Figure 3 shows the pseudo code of the E-MAP algorithm. There are three main steps of E-MAP, the first step is the initialization process and second and third mines all the maximal complex periodic patterns. The reason behind why we separate the second and third step is explained later in this section. In step one of the algorithm all the *Event-Block-Memories*, *Period-Block-Memories* and *Offset-Segment-Memories* are created. In the same step all the *Start_Occurance* and *Repetition* components of *Offset-Segment-Memories* are set to 0 and *Last_Occurance* are set

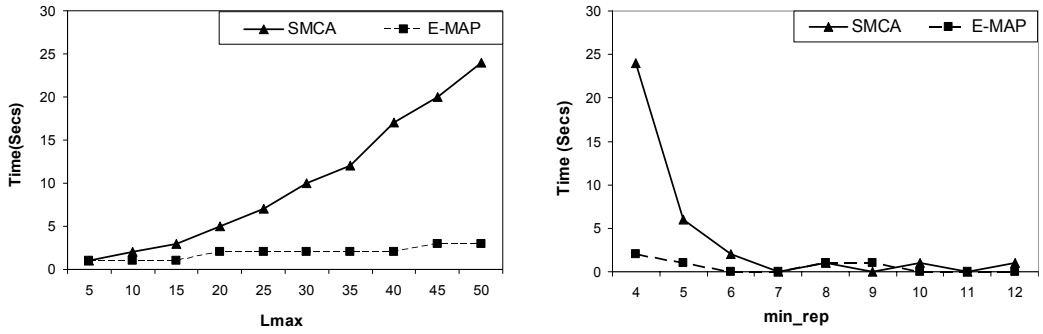
to *-1*, lines from 1 to 5 in Figure 3 show the pseudo code this step. In the second step of E-MAP, the algorithm first scans each time instance *⟨t⟩* of dataset with all of its events *⟨e⟩* one by one, lines from 6 to 25 in Figure 3 show this process. Next, the algorithm first calculates the offset of *⟨t⟩* with all time periods *⟨P⟩* in L_{max} and modifies the *Start_Occurance*, *Last_Occurance* and *Repetitions* components of *Offset-Segment-Memories* of event *⟨e⟩* period *⟨P⟩* and offset *⟨t % P⟩* according to the Rule 1. After scanning each event *e* of time instance *t* the difference between the *Last_Occurance* of *OSmem⟨event = e, period = P, offset = t % P⟩* and current time instance *⟨t⟩* is evaluated for maximal complex period patterns using the following principles.

- If only a single *Offset-Segment-Memory* is modified in any particular *PBmem* of length *l*, and contains a *Repetition* component value greater than *min_rep* threshold, then the pattern is declared as single event of period length *l* and reference to that event which it belongs. For example if any *OSmem⟨event = Z, P = 2, offset = 1⟩.Repetition* contains a value greater than *min_rep* so this generates a single event pattern and reference to event *⟨Z⟩* of period length *2*.
- If more than one *Offset-Segment-Memories* are modified in any *Period-Block-Memories* but of different period lengths, and contain *Repetition* component values greater than *min_rep*. Then the *Offset-Segment-Memories* are declared as single event patterns of different period lengths and reference to those events from which they belongs. For example with *min_rep = 3*, if any two *OSmem⟨event = A, period = 1, offset = 0⟩.Repetition* and *OSmem⟨event = B, period = 2, offset = 1⟩.Repetition* contain values greater than *min_rep*, so both of these are single event patterns *(pattern = A and pattern = B)* with period length of *1* and *2* respectively.
- If more than one *Offset-Segment-Memories* of same offset size are modified in same *PBmem* of size *l* and contains a *Repetition* component value greater than *min_rep*. Then the pattern is declared as multiple events pattern of period length *l* and reference to those events from which they belongs. For example with *min_rep = 3*, if any two *OSmem⟨event = Y, period = 2, offset = 1⟩.Repetition* and *OSmem⟨ event = Z, period = 2, offset = 1⟩.Repetition* contain values greater than *min_rep* threshold, so these generates a multiple events pattern *(pattern = YZ)* of period length *2*.
- If more than one *Offset-Segment-Memories* of different offset sizes are modified in *PBmem* of length *l* but of same size, and contains a *Repetition* component values greater than *min_rep*. Then the pattern is declared as maximal complex pattern of period *l* and reference to those events from which they belongs. For example with *min_rep = 3*, if any two *OSmem⟨event = Y, period = 2, offset = 0⟩.Repetition* and *OSmem⟨event = Z, period = 2, offset = 1⟩.Repetition* contain a value greater than *min_rep* threshold so this generates a maximal complex pattern *(pattern = (Y, Z))* with period length *2*.

In step 2 of the algorithm only those *Offset-Segment-Memories* are evaluated for maximal complex patterns which are modified by the events of time instances *⟨t⟩*. However, some *Offset-Segment-Memories* are left unevaluated, thereby in step 3 of the algorithm the *Offset-Segment-Memories* of all *Period-Block-Memories* are evaluated for maximal complex patterns using the same principle same as we have described for the step 2. Lines from 26 to 33 in Figure 3 show this step.

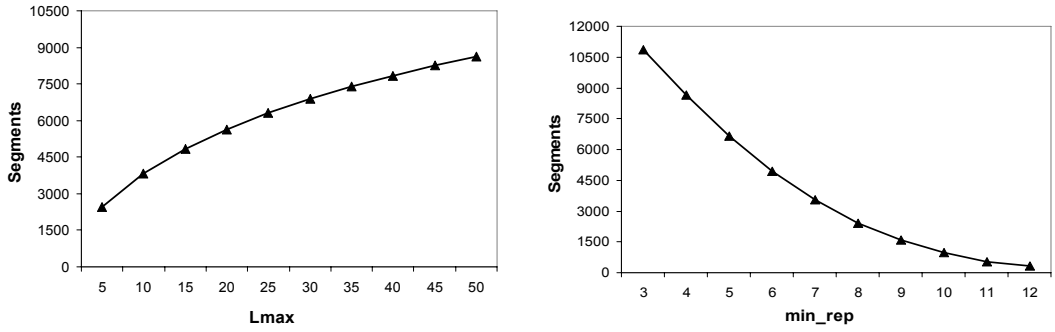
5. Computational Experiment

To evaluate the performance of E-MAP, we performed detailed experimental results on real and artificial generated large temporal datasets by varying the values of different factors to visualize the effect on performance. The experiments are performed on a computer 1.6 GHz CPU clock rate and 256 MB of main memory. All the source code of E-MAP and SMCA are written in Visual C++ 6.0. Our experimental dataset consists of the web log data of National University of Computer and Emerging Sciences Islamabad. The data was recorded mostly during 9 am to 8 pm. So for each day we filtered out the data for these particular hours only. The granularity was set to be one hour and data recorded in an hour was considered to be of a single time stamp. Furthermore we only considered the access patterns of 68 different clients IP. The data spanned over a time of 4 years starting from 2002. Approximately more than 3700 time instants were recorded in the data. Figure 4 shows the performance curve of E-MAP and SMCA on two different *min_rep* and L_{max} thresholds. As clear from Figure 4, the E-MAP algorithm outperforms SMCA on almost all levels of *min_rep* and L_{max} thresholds. Figure 5 shows the number of segments mined on these two different thresholds.



(a) **(b)**

Fig. 4. Performance results of E-MAP and SCMA on two different ***(a)*** L_{max} with *min_rep* = 5 ***(b)*** *min_rep* thresholds with L_{max} = 40.



(a) **(b)**

Fig. 5. Number of segments mined on two different ***(a)*** L_{max} with *min_rep* = 5 ***(b)*** *min_rep* thresholds with L_{max} = 40.

6. Conclusion

In this paper we proposed a novel algorithm E-MAP for mining asynchronous periodic patterns. Our algorithm mines all maximal complex patterns in a single step using single scan of temporal dataset without mining single event and multi events patterns. Our different extensive experiments on real as well as artificial generated temporal datasets show that mining maximal complex periodic patterns using E-MAP is more efficient and fast as compared to previous well known approach SMCA, which is 3 steps algorithm for mining maximal complex patterns. This shows the effectiveness of our approach.

7. Reference

[1] M. Antunes and L. Oliveira, "Temporal data mining: An Overview", In *Proc. of the KDD 2001 Workshop on Temporal Data Mining*, San Francisco, CA, USA, 2001.

[2] J. Yang, W. Wang and P. Yu, "Mining Asynchronous Periodic Patterns in Time Series Data", *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 613-628, May/Jun, 2003.

[3] K. Huang and C. Chang, "SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporal Databases", *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 774-785, Jun, 2005.

[4] B. Ozden, S. Ramaswamy and A. Silberschatz, "Cyclic association rules", In *Proc. of the 14th International Conference on Data Engineering*, pages 412–421, 1998.

[5] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements", In *Proc. of the 5th International Conference on Extending Database Technology*, volume 1057 Lecture *Notes in Computer Science*, Springer, 1996.

[6] J. Yang, W. Wang and P.S. Yu, "Mining asynchronous periodic patterns in time series data", In *Proc. Of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 275–279, 2000.

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.