Novel Algorithms for Asynchronous Periodic Pattern Mining Based on 2-D Linked List

Jieh-Shan Yeh[1*], Szu-Chen Lin[1] and Shueh-Cheng Hu[2]

[1]*Department of Computer Science and Information Management, Providence University, Taiwan*

[2]*Department of Computer Science and Communication Engineering, Providence University, Taiwan*

**(Corresponding Author) jsyeh@pu.edu.tw*

Abstract

Periodic pattern mining has gained a great attention in the past decade. Previous studies mostly focus on synchronous periodic patterns. The literature proposes many methods for mining periodic patterns. Nevertheless, asynchronous periodic pattern mining has gradually received more attention recently. In this paper, we propose an efficient 2-D linked structure and the ***OEOP*** *(One Event One Pattern) algorithm to discover all kinds of valid segments in each single event sequence. Then, referring to the general model of asynchronous periodic pattern mining proposed by Huang and Chang, this study combines these valid segments found by OEOP into 1-patterns with multiple events, multiple patterns with multiple events and asynchronous periodic patterns. The experimental results show that these algorithms have good performance and scalability.*

Keywords*: Periodic pattern, asynchronous sequence, data mining, pattern mining, sequential pattern*

1. Introduction

Pattern mining is an extensively studied topic in the research of data mining. Researchers have introduced and implemented various pattern-mining models for different applications. For transaction databases, there are *frequent itemset* mining [1, 2] and *sequential pattern* mining [3, 4]. For event sequence databases, there is *frequent episode* mining [5, 6, 7].

Periodic patterns commonly appear in all kinds of time-series databases. For instance, trajectories of objects, weather, tides, stock market prices, DNA sequences, etc. The discovery of patterns with periodicity is of great importance and has rapidly developed in recent years. The periodic pattern mining models include *full-cycle periodic pattern* mining [8], *segment-wise periodic pattern* mining [9], *partial periodic pattern* mining [10], *frequent partial periodic pattern* mining [10], and *asynchronous periodic pattern* mining [11, 12, 13 , 14, 15].

Yang, et al., [11] first proposed the concept of the asynchronous periodic patterns to deal with disturbances in data sequences. They aimed to discover the longest periodic subsequence that contains a small disturbance. To accelerate the mining process of discovering asynchronous periodic patterns, this study proposed an efficient linked list structure and the **OEOP** (One Event One Pattern) algorithm to discover all kinds of valid segments in each single event sequence. Afterwards, by calculating the offsets of the valid 1-

pattern segments, the proposed **MEOP** (Multiple Events One Pattern) algorithm and **MEMP** (Multiple Events Multiple Patterns) algorithm merged them into multiple-event patterns. Finally, the proposed **APP** (Asynchronous Periodic Patterns) algorithm produced asynchronous periodic patterns.

2. Problem Definition

This section defines the problem of asynchronous periodic pattern mining. The problem definition and notations are similar to [15] with minor modification.

Let $E=\{e_1,e_2,\ldots,e_n\}$ be a set of all events. An ***eventset*** X is a nonempty subset of E. An eventset with k events is called a k***-eventset***. A ***sequence*** D is an ordered list of eventsets. For example, $E=\{a,b,c,d\}$, $X=\{a,b,c\}$ is a *3*-eventset, $D=(\{a,b,c\}\{b,c\}\{a,c,d\}b\{a,c\}d\{a,b,c,d\}a\{a,c,d\}\{a,c\}d\{a,b,c,d\})$ is a sequence.

Definition 1. A ***pattern*** with period l is a nonempty sequence $P=(p_1,p_2,\ldots,p_l)$, where p_1 is an eventset and p_i is either an eventset or *, for $2\le i\le l$. The symbol * indicates a "don't care" position. A pattern P is called an ***i-pattern*** if exactly i positions in P contain eventsets. For example, $(\{a,b\},b,{*}c,{*})$ is a 3-pattern with period 5.

Definition 2. For two patterns $P=(p_1,p_2,\ldots,p_l)$ and $P'=(p_1',p_2',\ldots,p_l')$ with the same period *l*, P' *is a* **specialization** *of* P if and only if $p_i\subseteq p_i'$ or $p_i={*}$, for $1\le i\le l$. For example, let $P=(a,{*},c,{*})$, $P'=(\{a,b\},b,c,{*})$ is a specialization of P.

Definition 3. For pattern $P=(p_1,p_2,\ldots,p_l)$ with period l and a sequence of eventsets $D=(d_1,d_2,\ldots,d_l)$, we say that P ***matches*** D if and only if $p_i\subseteq d_i$ or $p_i={*}$, for $1\le i\le l$. D is also called a ***match*** of P. For example, let $P=(a,{*},c,{*})$, $D=(\{a,b\},b,\{a,b,c\},\{b,d\})$ is a match of P.

Consider pattern $P=(p_1,p_2,\ldots,p_l)$ with period *l*, a original sequence of eventsets $D=(d_1,d_2,\ldots,d_m)$ with length *m*, two subsequences $D_1=(d_i,d_{i+1},\ldots,d_{i+l-1})$ and $D_2=(d_j,d_{j+1},\ldots,d_{j+l-1})$ of D where $1\le i\le j\le m$:

If $i\le j\le i+l-1$, D_1 and D_2 ***overlap*** each other .

If $i+l-1<j$, the ***distance*** of D_1 and D_2 is $j-(i+l-1)+1$.

Definition 4. Given a pattern P with period *l*, a original sequence D, and k subsequences $D_1,D_2,\ldots,D_k$ of D, if D_i ($1\le i\le k$) matches P and the distance of D_i and D_{i+1} ($1\le i\le k-1$) equals 0, the sequence $D_1D_2\ldots D_k$ is called a ***k-segment*** (or a ***continuous matching block*** with the ***repetition*** *k*) of P. For example, let $P=(a,{*},c,{*})$, $S=(a,b,c,d,\{a,b\},b,\{a,b,c\},\{b,d\},a,a,c,c)$ is a 3-segment of P, since P matches $D_1=(a,b,c,d)$, $D_2=(\{a,b\},b,\{a,b,c\},\{b,d\})$, and $D_3=(a,a,c,c)$.

Definition 5. A maximum segment S with respect to a pattern P is called a ***valid segment***, if and only if the number of repetitions of S is no less than a given ***minimum repetition*** (i.e., *min_rep*). For example, let $P=(a,{*},c,{*})$ and *min_rep=3*, $S=(a,b,c,d,\{a,b\},b,\{a,b,c\},\{b,d\},a,a,c,c)$ is a valid segment w. r. t. P.

Problem Definition. Given a sequence of eventsets *D*, a minimum repetition *min_rep*, a maximum distance *max_dis*, an asynchronous periodic pattern *P* indicates that there exists a valid subsequence *S* with respect to *P* in *D* and *S* is a set of non-overlapping valid segments, where each valid segment has at least *min_rep* contiguous matches of *P* and the distance between any two successive valid segments does not exceed *max_dis*. ***Asynchronous periodic pattern mining*** (**APPM**) discovers all asynchronous periodic patterns in *D*.

3. Proposed Data Structures and Algorithms

Figure 1 illustrates the steps of the proposed mining process for asynchronous periodic pattern mining.

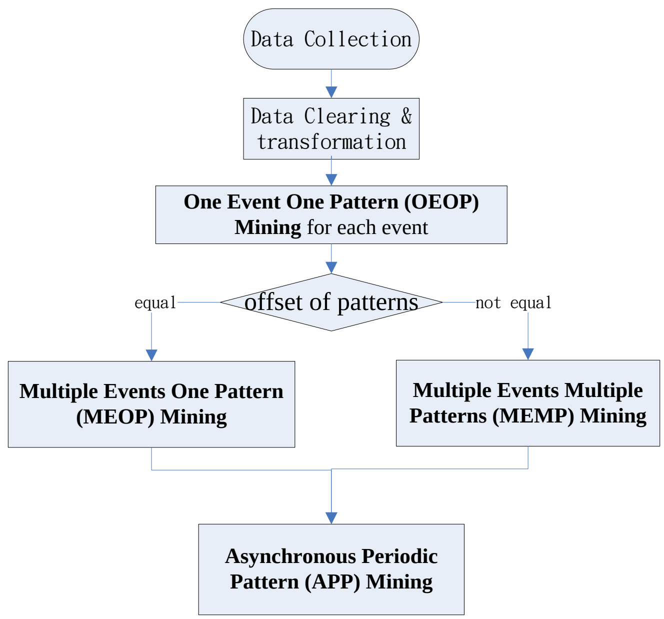

Figure 1. Proposed Process for APPM

To accelerate the mining process and properly record the pattern information of the list of time instants, we introduce the following three structures, **START** node, **END** node, and **VALID** node. By efficiently connecting **START** nodes and **END** nodes while processing the time instants, we are able to obtain all 1-patterns for the given event for its list of time instants.

START node: A structure consists of three fields. The first field, *stime*, saves the starting time instant of a 1-pattern; the second field, *next_s*, is a pointer that links to the next START node; the third field, *list_e*, is a pointer linking to an END node.

END node: A structure consists of four fields. The first field, *etime*, saves an ending time instant of a 1-pattern; the second field, *period*, records the period of the pattern; the third field, *rep_num*, stores the repetition of the pattern; the last field, *next_e*, is a pointer that links to the next END node.

VALID node: A 4-field structure to record a valid 1-pattern. The fields, *stime*, *etime*, *period*, and *rep_num*, indicate the starting time instant, the ending time instant, the period, and the repetition of the 1-pattern, respectively.

The structures of **START, END,** and **VALID** nodes are shown in Figure 2.

stime	next_s	list_e		START structure
etime	period	rep_num	next_e	END structure
stime	etime	period	rep_num	VALID structure

Figure 2. The Linked List Structures

3.1 OEOP (One Event One Pattern Mining) Algorithm

Given a sequence of eventsets *D*, for each event *e*, we first generate the list of time instants of *e*, denoted as TL_e. The preliminary goal of **OEOP** is to discover all valid 1-pattens in TL_e.

For each list of time instants TL_e of event *e*, with the minimal repetition *min_rep*, and the maximal period *Lmax*, the **OEOP** algorithm utilizes the new linked list structures and generates all valid 1-pattern segments of event *e*. The details of **OEOP** are as follows:

```
OEOP Algorithm
Input: the list of time instants TL_e for event e, min_rep, Lmax
Output: valid segments VS of event e
Method:
L= null ;
     //L : the list of Start node allocate a valid array VS
for each time instant t in TL_e do
{
     allocate a START node X;
    X.stime = t ;
    X.next = null;
    X. list_e = null ;
    L.insert(X) ;      // insert X at the end of L
    for each X_i node L do
    {
      for each Y_j node in X_i. list_e do
      {
        if ( t -Y_j.etime = Y_j.period )
        Y_j.etime = t ;   Y_j.rep_num++ ;
        if ( t -Y_j.etime > Y_j.period )
          {
           if ( Y_j.rep_num >= min_rep )
                move VS(Y_j) ;
```

```
                 //insert Yj at the end of VS array
              free (Yj) ;  // delete Yj
         }
       if ( t-Xi.stime <= Lmax )
         {
            allocate END node Y;
          Y.etime = t ;
          Y.period = t -Xi.stime ;
         Y.rep_num = 2 ;
          Xi.list_e.insert (Y) ;
          // insert Y at the end of Xj.list_e
         }
    }
}
return VS;
```

3.2 MEOP and MEMP Algorithms

After obtaining all valid segments of 1-patterns for each event, we record them in the following format: (*event*, *startTime*, *period*, *rep_num*). For example, (A, 1, 2, 4) indicates that the 1-pattern for event A starts at time 1, is during period 2, and repeats 4 times.

For valid segments of two different events with different starting times, the offsets of the two segments are calculated by the formula *offset*=*startTime* % *period*. Two segments with the same offset are possibly combined into a multiple-event segment.

The overlapping section of two valid segments is from min{e_i.*endTime*, e_j.*endTime*} to max{ e_i.*startTime*, e_j.*startTime*}, where *endTime*= *startTime*+ (*rep_num-1*) * *period*. If valid segments can be combined, we denote the result as:

$(\{e_1, \ldots, e_n\}, \max\{e_i.startTime\}, p, \lceil (\min(e_i.endTime) - \max\{e_i.endTime\}) / p \rceil + 1)$. For example, the combination of (A, 2, 2, 3) and (B, 2, 2, 3) is recorded as ({A, B}, 2, 2, 3), which is a multiple-event 1-pattern ({A, B} , *).

By computing the repetition of the overleaping section of two 1-patterns, **MEOP** (Multiple Event One Pattern) algorithm generates all valid 1-pattern segments with multiple events. The details of the algorithm are omitted here.

Alternatively, two segments with different offsets are possibly combined into a multiple-event multiple-pattern segment. Similarly, by computing the repetition of the overleaping section of two 1-patterns, the **MEMP** (Multiple Event Multiple Pattern) algorithm generates all valid multiple pattern segments with multiple events. The details of the algorithm are also omitted here.

3.3 APP Algorithm

After obtaining all valid patterns (single or multiple) of multiple events, with the minimal repetition *min_rep*, the maximal period *Lmax*, and the maximal distance of valid segments *max_dis*, the **APP** algorithm produces valid asynchronous segments with multiple events. The details of **APP** algorithm are as follows:

APP Algorithm

Input: *MVS*: array with patterns (single or multiple) of multiple events, *min_rep*, *Lmax* , *max_dis*

Output: *ASP_seq*: valid asynchronous segments with multiple events, in the format of (pattern, start time of segment$_1$, end time of segment$_1$, start time of segment$_2$, end time of segment$_2$, period)

Method:

for mvs_i and mvs_j in *MVS* with $mvs_i.stime > mvs_j.stime$ **do**
{
  if($0 < (mvs_i.etime - mvs_j.stime) <= max\_dis$ &&
      $mvs_i.period = mvs_j.period$) //non-overlap
    move *ASP_seq (pattern,* $mvs_i.stime$, $mvs_i.etime$, $mvs_j.stime$,
      $mvs_j.etime$, $mvs_j.period$ *)*
  if($0 > (mvs_i.etime - mvs_j.stime)$ && $mvs_i.period = mvs_j.period$) //overlap
  {
    $dis = | mvs_i.etime - mvs_j.stime| / l$;
    $des1 = mvs_i.etime - ((dis+1) * mvs_j.period)$; //forwardly shrinking
    $des2 = mvs_j.stime + ((dis+1) * mvs_j.period)$; //backwardly shrinking
    $rep1 = (des1 - mvs_i.stime) / mvs_j.period$;
    $rep2 = (mvs_j.etime - des2) / mvs_j.period$;
    if($rep1 >= min\_rep$) // forwardly shrinking with repeat >=*min_rep*
      move *ASP_seq (pattern,* $mvs_i.stime$, $des1$, $mvs_j.stime$,
        $mvs_j.etime$, $mvs_j.period$ *);*
    if($rep2 >= min\_rep$) // backwardly shrinking with repeat >=*min_rep*
      move *ASP_seq (pattern,* $mvs_i.stime$, $mvs_i.etime$, $des2$,
        $mvs_j.etime$, $mvs_j.period$ *);*
  }
return *ASP_seq;*

4. Experimental Results

4.1 Datasets

GenBank Sequences

By using the Entrez interface from the National Center for Biotechnology Information database, we randomly selected two protein genbank sequences with different data sizes. Figure 3 lists the first 1800 symbols in the sequence of the Trema virgata's genomic DNA (AJ131352). The symbols a, g, t, and c represent the purines adenine, guanine, pyrimidines thymine, and cytosine, respectively.

```
   1 atgagcagct cagaagttga caaagttttc
     acagaagagc tggaagctct ggtggtgaaa
  61 tcatgggctg taatgaagaa gaactctgct
     gaactgggtc ttaaattctt cctcaagtaa
 121 gtcaagatta tagatagtac actttttatt
     tactttgctt cttttgtaga ctaagttttt
```

Figure 3. AJ131352 GenBank Sequence

Stock Price Series

Second, we selected the 2008 Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX) by Taiwan Stock Exchange Co., Ltd. (TSEC) [16] and the 2008 Dow Jones Industrial Average Index ($INDU) by Dow Jones & Company [17]. Due to TSEC regulations, the daily change of TAIEX is limited to between -7% and 7%. Therefore, we transformed both TAIEX and $INDU numerical index series to the symbolic series using the following formula:

Change_rate(*i*-th day) = (*i*-th day's index – (*i*-*1*)-th day's index) / (*i*-*1*)-th day's index
Event(*i*-th day) = A, if Change_rate(*i*-th day)>=3%
Event(*i*-th day) = B, if 3% > Change_rate(*i*-th day)>= 1%
Event(*i*-th day) = C, if 1% > Change_rate(*i*-th day)>= -1%
Event(*i*-th day) = D, if -1% > Change_rate(*i*-th day)>= -3%
Event(*i*-th day) = E, if -3% > Change_rate(*i*-th day)

For example, in Table 1, the change rate of TAIEX on 2008/01/03 is (8,184.20-8,323.05)/ 8,323.05≒-0.01668. We set it to be the symbol D. In Table 2, the change rate of $INDU on 2008/01/03 is (13056.72328-13043.96091)/ 13043.96091≒-0.000978. We set it to be the symbol C.

Table 1. Example of TAIEX

Date	TAIEX	%	Event
2008/01/02	8,323.05		
2008/01/03	8,184.20	-0.01668	D
2008/01/04	8,221.10	0.004509	C
2008/01/07	7,883.37	-0.04108	E
2008/01/08	7,962.91	0.01009	B
2008/01/09	8,085.06	0.01534	B
2008/01/10	8,057.27	-0.00344	C

Table 2. Dow Jones Industrial Average Index ($INDU)

Date	$INDU	%	Event
2008/01/02	13043.96091		
2008/01/03	13056.72328	0.000978	C
2008/01/04	12800.17514	-0.01965	D
2008/01/07	12827.48825	0.002134	C
2008/01/08	12589.06756	-0.01859	D
2008/01/09	12735.30651	0.011616	B
2008/01/10	12853.09429	0.009249	C

Synthetic data for multiple eventsets

Both GenBank sequences and transformed stock price sequences only include one event at each time instant. For examining the performance of **MEMP** algorithm, we also artificially generated a multiple eventsets sequence, named AM_seq, from a randomly selected GenBank sequence. The basic information of each sequence investigated in the experiments are given in Table 3.

Table 3. Basic Information of Sequences

Sequence	Length	Event (count)
AJ131352	1104	a:331, t:363, g:217, c:191
X60729	1615	a:474, t:467, g:367, c:307
2008 TAIEX	248	A:16, B:44, C:111, D: 51, E:26
2008 $INDU	252	A:18, B:41, C:119, D:20, E:20
AM_seq	694	A:191, B:331, C:199

4.2 Numbers of valid segments and sub-sequences

By applying the **OEOP** algorithm on the X60729 GenBank sequence, the 2008 TAIEX sequence and the 2008 $INDU sequence, we obtained valid 1-pattens. Then, by utilizing **MEMP** and **APP** algorithms, we generated valid sub-sequences. Tables 4-5 list the numbers of valid segments and valid sub-sequences for the X60729 GenBank sequence, the 2008 TAIEX sequence and the 2008 $INDU sequence with *min_rep*=3, *period*=3, and *max_dis*=4.

Table 4. Numbers of Valid Segments and Sub-sequences of X60729 GenBank

X60729	number of valid segments	number of valid sub-sequences
(a, *, *)	65	9
(t, *, *)	62	9
(g, *, *)	36	3
(c, *, *)	18	0
(a, g, *)	6	0
(a, *, g)	3	0
(t, g, *)	4	0
(c, t, *)	6	0

Table 5. Numbers of Valid Segments and Sub-sequences of 2008 TAIEX

2008 TAIEX	number of valid segments	number of valid sub-sequences
(B, *, *)	2	0
(C, *, *)	71	42
(D, *, *)	7	3
(C, *, D)	3	0

4.3 OEOP Results

Figure 4 compares *min_rep* with the number of valid segments. For both X60729 GenBank and 2008 TAIEX sequences, the number of valid segments varies almost as the inverse of *min_rep*. In Figure 5, as expected, the increase in the size of *min_rep* is observed with decreasing running time, for both X60729 GenBank and 2008 TAIEX sequences. Fig. 6 illustrates that the size of the pattern period is not clearly related to the number of valid segments for both X60729 GenBank and 2008 TAIEX sequences.

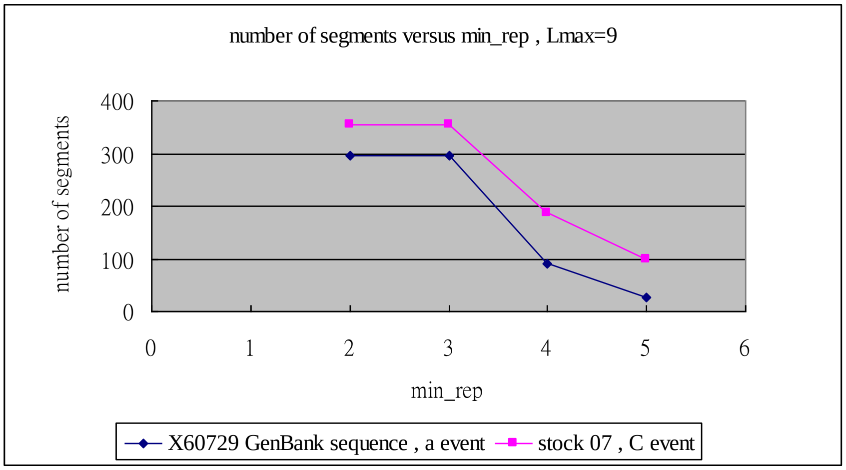

Figure 4. ***min_rep*** **vs Number of Valid Segments**

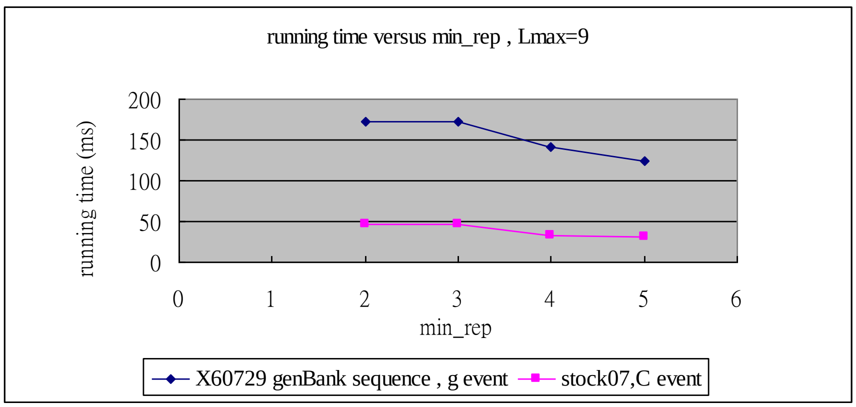

Figure 5. ***min_rep*** **vs Running Time**

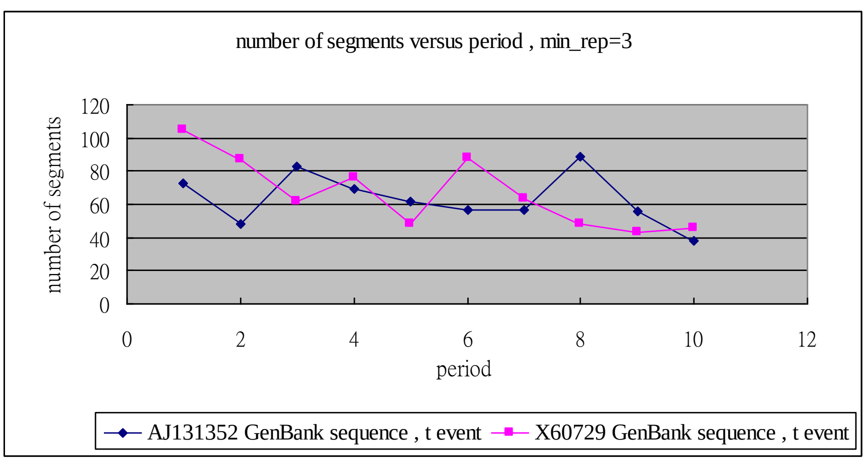

Figure 6. Period vs Number of Valid Segments

4.3 MEOP and MEMP Results

For the synthetic data sequence AM_seq, we calculated the numbers of segments including multi-event 1-patterns and multi-patterns by applying **MEOP** and **MEMP** algorithms. Fig. 7 demonstrates that the numbers of segments and *min_rep* size are inversely related for events {A, B}, {A, C} and {B, C}.

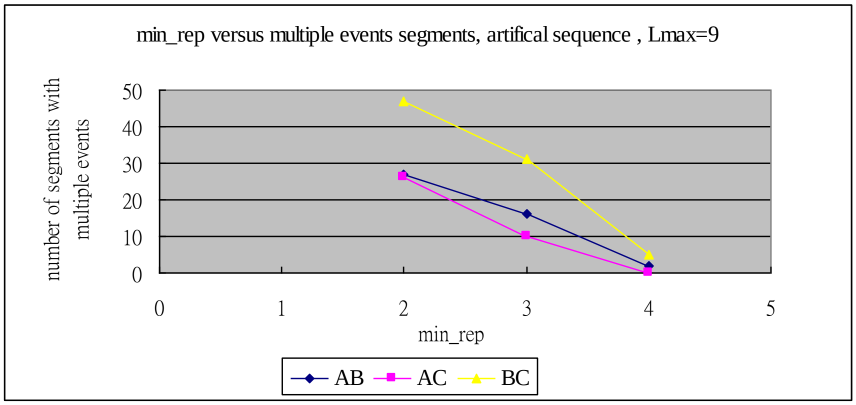

Figure 7. *min_rep* vs Number of Segments with Multiple Events

5. Conclusions

In this paper, we proposed an efficient linked list structure and **OEOP** algorithm to discover all kinds of valid segments in each single event sequence. The proposed **MEOP** and **MEMP** algorithms merge 1-patterns into multi-event 1-patterns or multi-event multi-patterns. Implementing these algorithms on real datasets, the experimental results show that these algorithms have good performance and scalability.

Acknowledgments

This work was supported by the National Science Council, Taiwan under grant NSC100-2221-E-126-016 and NSC99-2632-E-126-001-MY3.

References

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, In Proceedings of the 20th International Conference Very Large Data Bases (VLDB’94), **(1994)**, pp. 487-499.
[2] J. Han, J. Pei and Y. Yin, “Mining frequent patterns without candidate generation”, In Proceedings of ACM SIGMOD International Conference Management of Data (SIGMOD ’00), **(2000)**, pp. 1-12.
[3] R. Agrawal and R. Srikant, “Mining sequential patterns”, In Proceedings of the 11th International Conference Data Eng. (ICDE ’95), **(1995)**, pp. 3-14.
[4] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The PrefixSpan approach", IEEE Transactions on Knowledge and Data Engineering, vol. 16, **(2004)**, pp. 1424-1440.
[5] H. Mannila, H. Toivonen and A.I. Verkamo, “Discovering frequent episodes in sequences”, In Proceedings of the 1st International Conference Knowledge Discovery and Data Mining, **(1995)**, pp. 210-215.
[6] H. Mannila, H. Toivonen and A. I. Verkamo, “Discovering generalized episodes using minimal occurrences”, In Proceedings of the 2nd International Conference Knowledge Discovery and Data Mining, **(1996)**, pp. 146-151.
[7] H. Mannila, H. Toivonen and A. I. Verkamo, “Discovering frequent episodes in event sequences”, Data Mining and Knowledge Discovery, vol. 1, no. 3, **(1997)**, pp. 259-289.

[8] H. J. Loether and D. G. McTavish, “Descriptive and inferential statistics: an Introduction”, Allyn and Bacon, **(1993)**.
[9] J. Han, W. Gong and Y. Yin, “Mining segment-wise periodic patterns in time-related databases”, In Proceedings of the 4th ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD’98), **(1998)**, pp. 214-218.
[10] J. Han, G. Dong and Y. Yin, “Efficient mining partial periodic patterns in time series database”, In Proceedings of the 15th International Conference Data Eng. (ICDE ’99), **(1999)**, pp. 106-115.
[11] J. Yang, W. Wang and P. S. Yu, "Mining asynchronous periodic patterns in time series data", In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, **(2000)**, pp. 275-279.
[12] J. Yang, W. Wang and P. S. Yu, "Infominer: mining surprising periodic patterns", In Proceedings of ACM SIGKDD International Conference Knowledge Discovery and Data Mining (KDD’01), San Francisco CA, USA, **(2001)**, pp. 395-400.
[13] J. Yang, W. Wang and P. S. Yu, "InfoMiner+: mining surprising periodic patterns with gap penalties", In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), **(2002)**, pp. 725-728.
[14] J. Yang, W. Wang and P.S. Yu, “Mining asynchronous periodic patterns in time series data”, IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 3, **(2003)**, pp. 613-628.
[15] K. Y. Huang and C. H. Chang, "SMCA: a general model for mining asynchronous periodic patterns in temporal databases", IEEE Transactions on Knowledge and Data Engineering, vol. 17, **(2005)**, pp. 774-785.
[16] Taiwan Stock Exchange Capitalization Weighted Stock Index, http://www.twse.com.tw/en/trading/exchange/MI_5MINS_INDEX/genpage/Report201010/A12120101007.php?chk_date=2010/10/07, **(2010)** October.
[17] Dow Jones Industrial Average Index, http://www.djaverages.com/?view=industrial&page=index-data, **(2010)** October.