Efficient Mining Strategy for Frequent Serial Episodes in Temporal Database†

Kuo-Yu Huang and Chia-Hui Chang

Department of Computer Science and Information Engineering,
National Central University, Chung-Li, Taiwan 320
want@db.csie.ncu.edu.tw, chia@csie.ncu.edu.tw

Abstract. Discovering patterns with great significance is an important problem in data mining discipline. A serial episode is defined to be a partially ordered set of events for consecutive and fixed-time intervals in a sequence. Previous studies on serial episodes consider only frequent serial episodes in a sequence of events (called simple sequence). In real world, we may find a set of events at each time slot in terms of various intervals (called complex sequence). Mining frequent serial episodes in complex sequences has more extensive applications than that in simple sequences. In this paper, we discuss the problem on mining frequent serial episodes in a complex sequence. We extend previous algorithm MINEPI to MINEPI+ for serial episode mining from complex sequences. Furthermore, a memory-anchored algorithm called EMMA is introduced for the mining task.

†This work was sponsored by Ministry of Economic Affairs, Taiwan under grant 94-EC-17-A-02-S1-029.

1 Introduction

Mining significant patterns in sequence(s) is an important and fundamental issue in knowledge discovery, including sequential patterns, frequent episodes, frequent continuities and periodic patterns [1]. In these studies, discovering frequent *serial episodes* is a basic problem in sequence analyzing[4]. The goal of episode mining is to find relationships between events. Such relationships can then be used in an on-line analysis to better explain the problems that cause a particular event or predict future result. Serial episode mining has been of great interest in many applications, including internet anomaly intrusion detection [2], biomedical data analysis and web log analysis.

The task of mining frequent episodes was originally defined on "a sequence of events" where the events are sampled regularly as proposed by Mannila et al. [4]. Informally, an episode is a partially ordered collection of events occurring together. The user defines how close is close enough by giving the width of the time window *win*. Mannila et al. introduced three classes of episodes. *Serial episodes* consider patterns of a total order in the sequence, while *parallel episodes* have no constraints on the relative order of event sets. The third class contains composite episodes like serial combination of parallel episodes.

Mannila et al. presented a framework for discovering frequent episodes through a level-wise algorithm, WINEPI [4], for finding parallel and serial episodes that are frequent enough. The algorithm was an Apriori-like algorithm with the "anti-monotone" property of episodes' support. The support of an episode is defined as the number of sliding windows, a block of *win* continuous records, in the sequence. Take the sequence $S = A_3A_4B_5B_6$ as an example, there are 6-3+3=6 sliding windows in S given $win = 3$, e.g. $W_1 = A_3$, $W_2 = A_3A_4$, $W_3 = A_3A_4B_5$, $W_4 = A_4B_5B_6$, $W_5 = B_5B_6$ and $W_6 = B_6$. Unfortunately, this support count has a defect in confidence calculation of an episode rule. For example, the serial episode rule "When event A occurs, then event B occurs within 3 time units" should have probability or confidence 2/2 in the sequence S since every occurrence of A is followed by B within 3 time units. However, since episode $< A >$ is supported by four sliding windows, while serial episode $< A, B >$ is matched by two sliding windows (W_3 and W_4), the above rule will have confidence 2/4.

Instead of counting the number of sliding windows that support an episode, Mannila et al. consider the number of minimal occurrences of an episode from another perspective. They presented MINEPI [3], an alternative approach to the discovery of frequent episodes from minimal occurrences (*mo*) of episodes. A minimal occurrence of an episode α is an interval such that no proper subwindow contains the episode α. For example, episode $< A >$ has *mo* support 2 (interval [3,3] and [4,4]), while episode $< A, B >$ has only *mo* support 1 from interval [4,5]. Thus, the above rule will have confidence 1/2. However, both measures are not natural for the calculating of an episode rule's confidence. Therefore, we need a measure that facilitates the calculation of such episode rules to replace the number of sliding windows or minimal occurrences.

In addition, we sometimes find several events occurring at one time slot in terms of various intervals, called *complex sequences*. Note that a temporal database is also a kind of complex sequence with temporal attributes. Mining frequent serial episodes in a complex sequence has more extensive applications than that in a simple sequence. Therefore, we discuss the problem on mining frequent serial episodes over a complex sequence in this paper, where the support of an episode is modified carefully to count the exact occurrences of episodes. We propose two algorithms in mining frequent episodes in complex sequences, including MINEPI+ and EMMA. MINEPI+ is modified from previous vertical-based MINEPI [3] for mining episodes in a complex sequence. MINEPI+ employs depth first enumeration to generate the frequent episodes by *equalJoin* and *temporalJoin*. To further reduce the search space in pattern generation, we propose a brand new algorithm, EMMA (**E**pisodes **M**ining using **M**emory **A**nchor), which utilizes memory anchors to accelerate the mining task. Experimental evaluation shows that EMMA is more efficient than MINEPI+.

Time	1	2	3	4	5	6	8	9	10	11	12	13	14	15	16
Events	A C	C	B D	A C	D E	B D	A C	B D	E	A C	B D	D E	A C	C E	B D

(a) A temporal database TDB

Time	1	2	3	4	5	6	8	9	10	11	12	13	14	15	16
ID	#1 #3 #5	#3	#2 #4 #6	#1 #3 #5	#4	#2 #4 #6	#1 #3 #5	#2 #4 #6		#1 #3 #5	#2 #4 #6	#4	#1 #3 #5	#3	#2 #4 #6

(c) Encoded horizontal database for TDB

ID	Item	Timelist
#1	A	1, 4, 8, 11, 14
#2	B	3, 6, 9, 12, 16
#3	C	1, 2, 4, 8, 11, 14, 15
#4	D	3, 5, 6, 9, 12, 13, 16
#5	A, C	1, 4, 8, 11, 14
#6	B, D	3, 6, 9, 12, 16

(b) Frequent itemsets for TDB

Fig. 1. Phase I and II for EMMA

2 Mining Serial Episodes

2.1 MINEPI+

MINEPI is an iteration-based algorithm which adopts breadth-first manner to enumerate longer serial episodes from shorter ones. However, instead of scanning the temporal database for support counting, MINEPI computes the minimal occurrences mo of each candidate episode from the mo of its subepisode by temporal joins. For example, we want to find all frequent serial episodes from a simple sequence $S = A_1A_2B_3A_4B_5$ with $maxwin = 4$ and $minsup = 2$. MINEPI first finds frequent 1-episode and records the respective minimal occurrence, i.e. $mo(A) = \{[1,1],[2,2],[4,4]\}$, $mo(B) = \{[3,3],[5,5]\}$. Using temporal join which connects events from different time intervals (less than $maxwin$), we get intervals [1,3], [2,3], [2,5] and [4,5] for candidate 2-tuple episode <A,B>. Since $[1,3]$ and $[2,5]$ are not minimal, the minimal occurrences of <A,B> will be $\{[2,3],[4,5]\}$. If we want to count the number of sliding windows that match serial episode <A,B>, interval $[1,3]$ should be retained since the first subwindow contains A. Therefore, we have support count 3 for serial episode <A,B> since [2,3] and [2,5] denote the same sliding window. To extend MINEPI for our problem, we also need equal join which connects events at the same interval for dealing with complex sequences. We will use these intervals to compute the right support count for the problem.

Given the maximum window bound $maxwin$, the **bound list** of a serial episode $P = < p_1, \ldots, p_k >$, is the set of intervals $[ts_i, te_i]$ $(te_i - ts_i < maxwin)$ such that $p_1 \subset X_{ts_i}$, $p_k \subset X_{te_i}$ and $[X_{ts_i+1}, X_{ts_i+2}, \ldots, X_{te_i-1}]$ is a supersequence of $< p_2, \ldots, p_{k-1} >$. Each interval $[ts_i, te_i]$ is called a matching bound of P. By definition, the bound list of an event Y is the set of intervals $[t_i, t_i]$ such that X_{t_i} supports Y. Given a serial episode $P =< p_1, \ldots, p_k >$ and a frequent 1-pattern f and their matching bound lists, e.g., $P.boundlist = \{[ts_1, te_1], \ldots, [ts_n, te_n]\}$ and $f.boundlist = \{[ts'_1, ts'_1], \ldots, [ts'_m, ts'_m]\}$. The operation **equal join** of P and f which computes the bound list for a new serial episode $P_1 =< p_1, \ldots, p_k \bigcup f >$ (denoted by $P \odot f$) is defined as the set of intervals $[ts_i, te_i]$ such that $te_i = ts'_j$ for some j $(1 \leq j \leq m)$. Similar to equal join, the operation **temporal join** (concatenation) of P and f (denoted by $P \cdot f$) which computes the bound list for new serial episode $P_2 =< p1, \ldots, p_k, f >$ is defined

as the set of intervals $[ts_i, te'_j]$ such that $te'_j - ts_i < maxwin$, and $te'_j > te_i$ for some j $(1 \leq j \leq m)$.

Different from MINEPI, we apply depth-first enumeration to pattern generation for memory saving. This is because breadth first enumeration must keep track of records for all episodes in two consecutive levels, while depth-first enumeration needs only to keep intermediate records for episodes generated along a single path. Note that MINEPI+ does't search the minimum occurrence in the temporal database, we call our algorithm as MINEPI+ since the vertical-based operation in MINEPI+ is similar to MINEPI. Though the extension of MINEPI discover all frequent serial episodes, MINEPI+ has the following drawbacks: **1. A huge amount of combinations**: Let $|I|$ be the number of frequent 1-episodes, WINEPI+ needs $|I|^2$ and $\frac{|I|^2-|I|}{2}$ checking for temporal joins and equal joins, respectively. **2. Unnecessary joins**: For example, while the number of the extendable matching bounds for a serial episode is less than $minsup * |TDB|$, we can skip all temporal joins for this prefix. **3. Duplicate joins**: For example, to find serial episode <ABC, ABC>, MINEPI+ needs four of equal joins (twice (<A>, <B>) and (<AB>,<C>)) and one temporal joins ((<ABC>, <A>)). However, if we maintain the bound list for $< ABC >$, we only needs one temporal joins.

2.2 EMMA

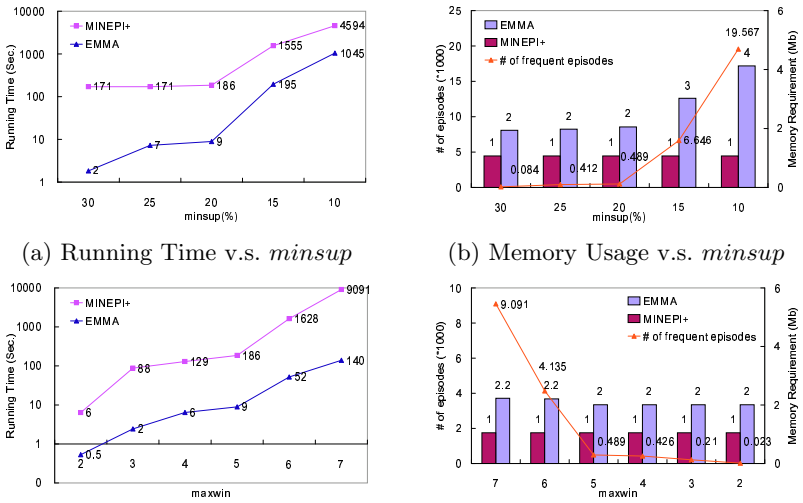
In this section, we propose an algorithm, EMMA (**E**pisode **M**ining using **M**emory **A**nchor), that overcomes the drawbacks of the MINEPI+ algorithm. To reduce duplicate checking, EMMA is divided into three phases, including (I) Mining frequent itemset in the complex sequence. (II) Encode each frequent itemset with a unique ID and construct them into a encoded horizontal database. (III) Mining frequent serial episodes in the encoded database. The EMMA algorithm adopts DFS to enumerate local frequent patterns by memory anchors to accelerate the mining task, which is more like a pattern growth method since it searches the local frequent sub-pattern to form the long pattern. Thus, instead of frequent items, we have a larger set of all frequent itemsets as frequent 1-tuple episodes. Again, we will use the boundlists for each frequent 1-tuple episode to enumerate longer frequent episodes. However, we only combine existing episodes with a "local" frequent 1-tuple episode to overcome the huge amount of candidate generation.

Now, in order to discover local frequent 1-tuple episode efficiently, we construct an encoded database EDB indexed by time (Phase II) and utilize the boundlists as a memory anchor to access the horizontal-based information. Note that the timelists of the frequent itemsets are equivalent to the boundlists for frequent 1-tuple episodes. As an example, Figure 1 shows an illustrative transaction database, the frequent itemsets with $min\_sup = 5$, and the encoded database. Finally, we use depth first enumeration to enumerate frequent serial episodes and carefully avoid unnecessary joins in Phase III.

Similar to MINEPI+, it adopts depth first enumeration to generate longer serial episodes. However, EMMA generates only frequent serial episodes by join-

ing an existing serial episode with local frequent IDs. This is accomplished by examining those transactions following the matching bounds of current serial episodes. For example, if we want to extend an episode #3={C} with boundlist {[1,1], [2,2], [4,4], [8,8], [11,11], [14,14], [15,15]}, we need to count the occurrences of IDs in the following intervals within $maxwin = 4$ bound, i.e. [2,4], [3,5], [5,7], [9,11], [12,14], [15,16] and [16,16]. We call these intervals the projected boundlist of a serial episode <#3>. Formally, the projected bound list of a boundlist for an episode is defined as follows. Given the bound list of a serial episode P, $P.boundlist = \{[ts_1, te_1], \ldots, [ts_n, te_n],\}$ in the encoded database ED, the **projected boundlist** (PBL) of P is defined as $P.PBL = \{[ts'_1, te'_1], \ldots, [ts'_n, te'_n],\}$ where ts'_i=$min(ts_i + 1, |TDB|)$ and $te'_i = min(ts_i + maxwin - 1, |TDB|)$.

When examining the IDs in the projected boundlist, we also record the boundlists of IDs. For example, #4 is a local frequent ID in #3.PBL and has boundlist {[3,3], [5,5], [6,6], [9,9], [12,12], [13,13], [16,16]}. Thus, when new serial episodes <C,D> are generated by temporal join <#3,#4>, we know their boundlists immediately, i.e. {[1,3], [2,3] [4,5], [8,9], [11,12], [14,16], [15,16]}. To extend this episode, the procedure *emmajoin* is called recursively until no more new serial episodes can be extended, i.e. when the number of extendable bounds for a serial episode is less than $minsup * |TDB|$. For example, suppose the boundlist of of some serial episode is {[1,3], [3,5], [8,11], [11,14], [14,15]}, with $maxwin = 4$ the extendable bounds include {[1,3], [3,5], [14,15]} since [8,11] and [11,14] already reach the maximum window bound. With $minsup = 5$, we do not need to extend serial episode <B>. This strategy can avoid unnecessary checking spent in MINEPI+.



(a) Running Time v.s. *minsup*

(b) Memory Usage v.s. *minsup*

(c) Running Time v.s. *maxwin*

(d) Memory Usage v.s. *maxwin*

Fig. 2. Performance comparison in real data

3 Experiments

We apply MINEPI+ and EMMA to a data set composed of 10 stocks in the Taiwan Stock Exchange Daily Official list for 2618 trading days from September 5, 1994 to June 21, 2004. We discretize the stock price of go-up/go-down into five levels. Figure 2(a) shows the running time with an increasing support threshold, *minsup*, from 10% to 30%. Figure 2(c) shows the same measures with varying *maxwin*. As the *maxwin*/*minsup* threshold increases/decreases, the gap between MINEPI+ and EMMA in the running time becomes more substantial. Figures 2(b) and (d) show the memory requirements and the number of frequent episodes with varying *minsup* and *maxwin*. As the *maxwin* threshold increases or *minsup* threshold decreases, the number of frequent episodes also increases. The memory requirement in MINEPI+ is steady. However, EMMA needs to maintain more frequent itemsets as the *minsup* decreases; whereas the memory requirement with varying *maxwin* in EMMA is changed slightly. Overall, MINEPI+ is better than EMMA in memory saving (by a magnitude of 4 for $minsup = 10\%$).

4 Conclusion and Future Work

In this paper, we discuss the problem of mining frequent serial episodes in a complex sequence and propose two algorithms to solve this problem. First, we modify previous vertical-based MINEPI to MINEPI+ as the baseline for mining episodes in a complex sequence. To avoid the huge amount of combinations/computations and unnecessary/duplicate checking, we utilize memory to propose a brand-new memory-anchored algorithm, EMMA. The experiments show that EMMA is more efficient than MINEPI+. So far we have only discussed serial episodes. Parallel episodes, which have no constraint on event orders, and composite episodes, e.g. serial combination of parallel episodes, remain to be solved. Thus, further researches are required.

References

1. K. Y. Huang and C. H. Chang. Smca: A general model for mining synchronous periodic pattern in temporal database. *IEEE Transaction on Knowledge and Data Engineering (TKDE)*, 17(6):776–785, 2005.
2. Jianxiong Luo and Susan M. Bridges. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, 15(8), 2000.
3. H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146–151, 1996.
4. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215, 1995.