2017 IEEE 33rd International Conference on Data Engineering

# Mining Precise-positioning Episode Rules from Event Sequences

Xiang Ao [#,1], Ping Luo [#,2], Jin Wang [†,3], Fuzhen Zhuang [#,1], Qing He [#,1]

[#]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing 100190, China
[†]Computer Science Department, University of California, Los Angeles.
[1] {aox,zhuangfz,heq}@ics.ict.ac.cn [2] luop@ict.ac.cn [3] jinwang@cs.ucla.edu

***Abstract*—Episode Rule Mining is a popular framework for discovering sequential rules from event sequential data. However, traditional episode rule mining methods only tell that the consequent event is likely to happen within a given time intervals after the occurrence of the antecedent events. As a result, they cannot satisfy the requirement of many time sensitive applications, such as program security trading due to the lack of fine-grained response time. In this study, we come up with the concept of *fixed-gap episode* to address this problem. A fixed-gap episode consists of an ordered set of events where the elapsed time between any two consecutive events is a constant. Based on this concept, we formulate the problem of mining *precise-positioning episode rules* in which the occurrence time of each event in the consequent is clearly specified. In addition, we develop a trie-based data structure to mine such precise-positioning episode rules with several pruning strategies incorporated for improving the performance as well as reducing memory consumption. Experimental results on real datasets show the superiority of our proposed algorithms.**

## I. Introduction

Frequent episode mining (FEM) has emerged as a popular research topic in the data mining community. Given a single event sequence, FEM aims to identify all frequent episodes with the frequency larger than a given threshold. Here an *episode* (also known as serial episode [3]) is a totally ordered set of events. One basic problem in FEM is to find episode rules from frequent episodes. Given a frequent episode $\alpha$, a valid episode rule in the form of $lhs \rightarrow rhs$ can be generated in a straightforward manner: The antecedent $lhs$ is the prefix of $\alpha$ and the consequent $rhs$ is the last event in $\alpha$, if its confidence is larger than a user-specified threshold.

Figure 1 gives a running example for episode rule mining, where capital letters denote events and arabic numbers denote timestamps. In this sequence, we see three occurrences of episode $\langle$D, A, B$\rangle$ when maximum occurrence window size threshold is set to 4. Then, if we take B as the consequent, an episode rule $\langle$D, A$\rangle \rightarrow \langle$B$\rangle$ can be generated. This rule tells that it is within 2 time intervals after the occurrence of $\langle$D, A$\rangle$ that B will occur (with 100% probability). However, with such a rule discovered by traditional methods, we only know the approximate time range of the occurrence of $rhs$. In many real world applications, such rules are not practically useful without specifying the exact time of $rhs$. In this situation, we require the exact time to trigger the responses of the consequent automatically.

In this paper, we study the problem of **Mi**ning **P**recise-positioning **E**pisode **R**ules (MIPER) where we need to specify the exact time instead of the approximate time range of the consequent events. This precise-positioning episode rule mining problem is motivated by some time-sensitive applications. One example application is program security trading. Suppose

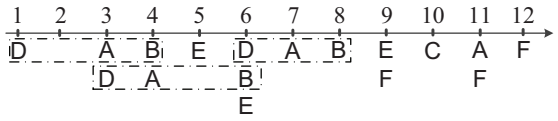



Fig. 1. The running example of event sequence.

the sequence in Figure 1 describes daily change events of a stock market, and B refers to an event depicting the price of a stock increases. In the above example, we cannot decide the exact time to buy this stock after the occurrence of $\langle$D, A$\rangle$. One choice is that we buy the stock at the first day and hold it for two days after $\langle$D, A$\rangle$ happens. However, we might lose money if the stock slumps significantly on the first day and rebounds slightly (B occurs) on the next day. Here though this rule makes a correct prediction, we would still lose money.

To address this problem, we first introduce the concept of *fixed-gap episode*, which is defined as a tuple of events such that the time span between any two consecutive events is specified. Formally, the goal is to mine a *precise-positioning episode rule* $lhs \xrightarrow{\Delta t} rhs$ with the following two constraints: 1) $rhs$ is a fixed-gap episode; 2) the elapsed time between the last event in $lhs$ and the first event in $rhs$ is $\Delta t$.

While precise-positioning episode rules could provide richer information, the performance of mining such rules would suffer from "combination explosion". Due to the constraints of exact time, the number of candidates for precise-positioning episode rules would be much larger than that of traditional episode rules.

In this paper, we first propose an enumeration based framework by first mining frequent minimal-occurrence episodes and frequent fixed-gap episodes on the whole sequence, and then concatenating each pair of a minimal-occurrence episode and a fix-gap episode to generate a candidate of precise-positioning rule. We further observed that for a precise-positioning episode rule, the consequent must occur after the antecedent. Hence we can improve the proposed framework by mining fixed-gap episodes only after the occurrences of frequent minimal-occurrence episodes. Along this route, we

2375-026X/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDE.2017.45

develop a trie-based framework to mine precise-positioning episode rules directly based on frequent minimal-occurrence episodes partitioning.

The major contributions of this work include:

- We address the new problem of mining precise-positioning episode rules to satisfy the requirement of time-sensitive applications in the real world.
- We design a trie-based framework to compactly store valid precise-positioning episode rules and perform efficient mining with effective pruning strategies.
- We demonstrate the efficiency of the proposed algorithms based on real-world datasets and the effectiveness of precise-positioning episode rules on practical application.

## II. Definitions and Problem Statement

In this section, we propose some new concepts about precise-positioning episode rules, and formulate the mining problem.

**Definition 1** (**Precise-positioning Episode Rule**). A *precise-positioning episode rule* (PER) $\Gamma$ is an implication of the form $\alpha \xrightarrow{\Delta t} \beta$, where $\alpha{=}\langle e_{\alpha_1}, \cdots, e_{\alpha_l}\rangle$ is a frequent minimal occurrence based episode, and $\beta{=}(\langle e_{\beta_1}, \cdots, e_{\beta_k}\rangle, \langle \Delta t_1, \cdots, \Delta t_{k-1}\rangle)$ is a fixed-gap episode. Also, it must satisfy the following two conditions: (1) the elapsed time between $e_{\alpha_l}$ and $e_{\beta_1}$ is a fixed value $\Delta t$; (2) For a given threshold $\epsilon$, $\sum_{i=1}^{k-1} \Delta t_i + \Delta t \leq \epsilon$. Here, $\epsilon$ is a user-specified threshold named *maximum window size for consequent occurrence*.

**Definition 2** (**Occurrence of PER**). Given a precise-positioning episode rule $\Gamma{=}\alpha \xrightarrow{\Delta t} \beta$, a minimal occurrence [4] (MEO for short) of $(\alpha, [t_{\alpha_1}, t_{\alpha_p}])$, and a Fixed-gap Episode Occurrence (FEO for short) of $(\beta, [t_{\beta_1}, t_{\beta_q}])$, we call $[t_{\alpha_1}, t_{\alpha_p}, t_{\beta_1}, t_{\beta_q}]$ the *occurrence* of $\Gamma$ if and only if $t_{\beta_1} - t_{\alpha_p} = \Delta t$. This occurrence is denoted as $(\Gamma, [t_{\alpha_1}, t_{\alpha_p}, t_{\beta_1}, t_{\beta_q}])$, and $t_{\alpha_1}$ and $t_{\beta_q}$ are called *start time* and *end time*, respectively. The set of all occurrences of $\Gamma$ is denoted as $\mathrm{ocSet}(\Gamma)$.

**Definition 3** (**Valid PER**). The *support* of PER $\Gamma{=}\alpha \xrightarrow{\Delta t} \beta$, denoted as $\mathrm{sp}(\Gamma)$, is defined as the number of its distinct occurrences. A PER is *valid* if its confidence, $\frac{\mathrm{sp}(\Gamma)}{\mathrm{sp}(\alpha)}$, is not less than a user-specified minimum confidence threshold $min\_conf$.

**Problem statement of MIPER**. Given an event sequence $\vec{S}$, the problem of MIPER is to find all valid PER over $\vec{S}$.

## III. The Solution to MIPER

In this section, we propose several solutions to MIPER. Here we always set the parameters as $min\_sup = 3$, $\epsilon = 5$ and $min\_conf = 0.6$ to show the running example in this section unless otherwise specified.

### A. The Enumeration Approach

The most intuitive way to solve the MIPER problem is an enumeration based approach, denoted as MIP-ENUM.

The basic idea of MIP-ENUM is to enumerate PER candidates by concatenating discovered MEOs with FEOs and subsequently filter the infrequent ones according their confidence values. By indexing the FEO and MEOs using their start and end times, we can form a PER occurrence by concatenating contiguous MEO and FEO thus generate a PER candidate. Finally, the filtering step is carried out to output all valid PERs.

In this paper, we design a level-wise method which mines fixed-gap $k+1$-episodes based on the frequent fixed-gap $k$-episode. The main idea of such approach is that for every frequent fixed-gap $k$-episode $\beta$, every occurrence of $\beta$ is considered independently. For each FEO $(\beta, [t_{\beta_1}, t_{\beta_k}])$, we scan each event set $E_i$ on $\vec{S}$ where $i \in [t_{\beta_k}+1, t_{\beta_1}+\epsilon-t_{\beta_k}]$ and generate new fixed-gap $k+1$-episode occurrences. After that, we filter infrequent fixed-gap $k+1$-episodes and keep the frequent ones for the next iteration.

### B. The Trie-based Approaches

The enumeration approach is very time consuming due to the large search space. Here we introduce a more efficient approach, named MIP-TRIE. In MIP-TRIE, we design a trie-based framework, named PER-trie, to store precise-positioning episode rules compactly. The features of PER-trie are also helpful to facilitate the efficiency of discovering valid PERs.

*1) Structure of PER-trie:* Given a frequent minimal-occurrence episode $\alpha$, a *PER-trie*, denoted as $\mathcal{T}_\alpha$ is a trie-like data structure which stores valid precise-positioning episode rules whose antecedent is $\alpha$.

The root node $r$ of a PER-trie $\mathcal{T}_\alpha$, denoted by $(r.episode{:}r.tlist)$, consists of two fields: the episode field $r.episode$ and the end time set field $r.tlist$. Here $r.episode$ registers the minimal-occurrence episode $\alpha$; and $r.tlist$ records all the end times of minimal occurrence of $\alpha$.

The non-root node $q$, denoted by $(q.event{:}q.tlist)$, consists of two fields: the event field $q.event$ and the occurrence time set field $q.tlist$. Here $q.event$ registers which event this node represents, and $q.tlist$ is a set containing the occurrence time of such event after a fixed *distance* to the elements of $p.tlist$, where $p$ is the parent node of $q$.

The edge between a parent node $p$ and its child node $q$ has a *distance* field, which is a positive integer to record the length between the two nodes. We denote it as $\mathrm{d}(p, q)$. In the PER-trie, for any element $t' \in q.tlist$, there exists an element $t \in p.tlist$ such that $t' - t = \mathrm{d}(p, q)$.

Then given a PER-trie $\mathcal{T}_\alpha$, a PER can be represented by a path from the root node to any non-root node. Specifically, we assume that the path from the root $r$ to a non-root node $q$ is through the nodes $q_1 \Rightarrow q_2, \ldots, \Rightarrow q_k$. Then, the node $q$ refers to a precise-positioning episode rule $\alpha \xrightarrow{d(r,q_1)} (\langle q_1.event, \ldots, q_k.event, q.event\rangle, \langle \mathrm{d}(q_1, q_2), \mathrm{d}(q_1, q_2), \ldots, \mathrm{d}(q_k, q)\rangle)$.

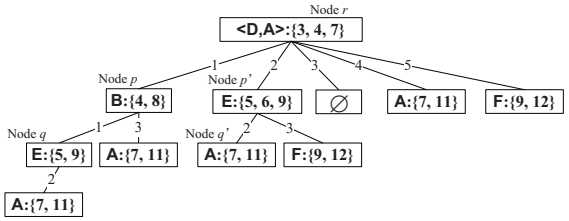



Fig. 2. Complete PER-trie of $\mathcal{T}_{\langle \mathsf{D},\mathsf{A}\rangle}$.

*2) Features of PER-trie:* Here we introduce two essential features of PER-trie. Such features will be helpful to further improve the performance of the mining process.

*Lemma 1:* [Support Counting] Given a non-root node $q'$ on a PER-trie $\mathcal{T}_\alpha$, the support of the PER associated with $q'$ is equal to the cardinality of the occurrence time set field of $q'$, i.e. $|q'.tlist|$.

*Lemma 2:* [Downward Closure] Given a pair of parent-child node $p$ (parent) and $q$ (child) in a PER-trie, the support of the rule associated with $p$ is no less than that of $q$.

*3) Mining Valid PER with PER-trie:* Here we introduce MIP-TRIE algorithm which directly discovers valid precise-positioning episode rules from the event sequence without generating PER candidates. The key ingredients to the efficiency of MIP-TRIE are (i) frequent minimal-occurrence episodes partitioning, (ii) the usage of PER-trie to store valid PERs and (iii) mining valid PERs with pruning strategies.

The MIP-TRIE algorithm is divided into two phases. The first phase is to mine frequent minimal-occurrence episodes as possible antecedents of PER. The second phase, subsequently, is to mine valid PERs based on each possible antecedent $\alpha$. Its output is a complete PER-trie that stores all valid PERs in the input sequence whose antecedent is $\alpha$.

MIP-TRIE benefits from the minimal-occurrence episode based partitioning. Compared with MIP-ENUM, MIP-TRIE only needs to discover corresponding fixed-gap episodes on subsequences after antecedents rather than the whole sequence, which alleviates the number of useless fixed-gap episodes.

MIP-TRIE in fact transfers the valid PER mining process to a *complete* PER-trie construction for each frequent minimal-occurrence episode. To perform efficient constructions, we adpot two different manners. The first one is DFS [2] manner (denoted as MIP-TRIE(DFS) specifically), and the other one, denoted as MIP-TRIE(PRU), is a novel hybrid manner with pruning techniques.

For MIP-TRIE(DFS), particularly, we start from the root node of a PER-trie, and recursively expand nodes on the PER-trie until no more nodes can be expanded. It firstly creates a root node $r$ with a given antecedent $\alpha$. Then, an internal function $\mathrm{ExtendNode}$ is invoked starting from $r$ to perform node expansions. For a node $q$ to be extended, it first calculates the distance $d(q, r)$. Then upper bound of the number of enumerations is correspondingly $\epsilon - d(q, r)$. Next, it conducts on multiple rounds of enumerations. Firstly, an offset value is appended to every element in $q.tlist$ to get a set of variables $position\_List$. All event sets occurring at time $position\_List$ are then collected. Among these event sets, it searches events with frequency higher than $min\_freq$ and store them to a set $E'$. For any event $e' \in E'$, it extends a new node $q'$ as the child of $q$, which represents a new discovered valid PER. Once $q'$ is generated, the same function is immediately invoked recursively in which $q'$ is taken as the node for the next extension. The construction process will stop when such an event set $E'$ cannot be found any longer.

Although MIP-TRIE(DFS) can significantly outperform MIP-ENUM by avoiding the generation of candidates, it may scan repetitive positions over the input sequence to collect event sets as well as their frequencies. In order to eliminate duplicate event sets and sequence scans, we propose MIP-TRIE(PRU) whose basic idea is to adjust the order of node extensions and reuse intermediate results of the established PER-trie as much as possible during the traversing process.

We give the pseudo-code of such method in Algorithm 1.It firstly expands the root node $r$ via the same way as that in MIP-TRIE(DFS) and then performs $\epsilon$ iterations in the outer loop. But unlike the DFS based method, when reaching a new child node (denoted by $q'$, of $r$), a top-down inner traverse procedure is triggered. That is, for every non-root node $w$ on the established PER-trie except that $q'$, it computes a variable set to record the time stamps that are needed to be checked (Line 13). Then if the cardinality of the intersection of such a variable set and $q'.tlist$ exceeds the minimum frequency threshold, a new child node will be directly generated to the node $w$, and a new valid PER can be derived (Line 14–17). Otherwise, all descendants of $w$ can be safely pruned according to the Lemma 2 (Line 19). Eventually, it will return a complete PER-trie after the outer loop is finished.

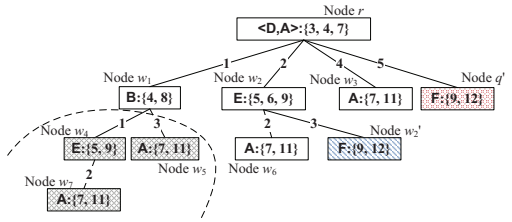



Fig. 3. The result for performing Alg. 1 to the PER-trie $\mathcal{T}_{\langle \mathsf{D},\mathsf{A}\rangle}$ when $i = 5$. Node $q'$ is firstly expanded. Node $w_1$–$w_3$ and $w_6$ are traversed and derive a new expanded node $w'_2$. Traverse on node $w_4$, $w_5$ and $w_7$ are pruned.

## IV. Experiments

We evaluate efficiency and effectiveness of the proposed algorithms. Experiments on efficiency are performed on a Linux server with 1.87GHz Intel Xeon E7-4807 CPU and 128GB memory. All of the algorithms are implemented in Java.

**Efficiency Evaluation:** The popular benchmark Retail [1] is used to evaluate the efficiency of the proposed algorithms. The compared methods include MIP-ENUM and two versions of MIP-TRIE since there is no prior work of PER mining. For mining frequent minimal-occurrence episodes, the recent MESELO [1] algorithm was adopted. We tuned parameters and demonstrate their impacts on performance. In each comparison, we vary one parameter while keep the others fixed.

Fig. 4 exhibits the results on time efficiency with varying parameters. From these results, we can observe MIP-TRIE series algorithms are significantly more efficient than

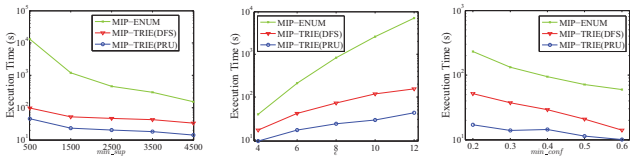
[1] http://fimi.cs.helsinki.fi/data/

**Algorithm 1**: MIP-TRIE(PRU)

**Input**: $\alpha$: a frequent minimal-occurrence episode
$ET_\alpha$: the set of end time of each MEO of $\alpha$
$\epsilon$: the threshold of maximum window size for consequent occurrence
$min\_freq$: minimum frequency threshold generating a valid PER
**Output**: $\mathcal{T}_\alpha$: the complete PER-trie w.r.t antecedent $\alpha$

```
1  begin
2      build a root node r of T_α where r.episode←α and r.tlist← ET_α
3      for i = 1 to ε do
4          position_List ← {t' | t' = t + i, t ∈ r.tlist}
5          scan S⃗ on each time stamp in position_List and get an event set
           E' such that the frequency of every event in E' is not less than
           min_freq
6          foreach e' ∈ E' do
7              q'.event ← e'
8              q'.tlist ← occurrence time of e' in position_List
9              put q' as the child of r
10             foreach non-root node w of T_α except q' do
11                 d ← distance between r and w
12                 if i > d then
13                     tmp_List={t' | t + i − d, t ∈ w.tlist}
14                     S ← tmp_List ∩ q'.tlist
15                     if |S| ≥ min_freq then
16                         w'.event ← q'.event, w'.tlist ← S
17                         put w' as the child of w
18                     else
19                         /* Pruning with Lemma 2*/
20 end
```

MIP-ENUM. Moreover, MIP-TRIE(PRU) outperforms MIP-TRIE(DFS) because of its improved trie construction strategy and powerful pruning techniques.




(a) $min\_sup$ (b) $\epsilon$ (c) $min\_conf$

Fig. 4. The efficiency comparison results.

**Effectiveness of PER:** Next we evaluate the effectiveness of PER on real sequences from China stock market. Here we consider 150 related industry sector pairs and discretize their daily index change ratios into event sequences. In more detail, we discretize the change ratio values into two categories and generate two kinds of events: UP (if the price increases) and DN (otherwise) for each sector. Finally, we obtain 150 event sequences. The data we used involves 1,129 trading days ranging from Jan. 1, 2010 to Aug. 29, 2014. We generate the training and test sets from this dataset as follows: The data from the first 4 years (including 967 trading days) are taken as the training set, and the rest (including 162 trading days) are used as the test set. Then we mine valid PER for every event sequence in the training set with the following parameter settings: $min\_sup$ = 145, $min\_conf$ = 0.5 and $\epsilon$ = 5. We empirically emphasize that compact rules are more convincing. It is difficult to believe a rule telling an industry sector may go up 15 days after viewing an antecedent happen.

Every sequence reported multiple valid PERs under our settings. Here we focus on demonstrate the difference of PER and traditional episode rules. Specifically, we degrade PER with single event in consequent to traditional episode rule (denoted as TDR hereafter) and demonstrate their differences in predictive ability. In order to show their differences more clearly, we collected such rules whose $\Delta t = 5$ from top 50 PER in the training sets ranking by confidence, and construct an experimental set. Since the consequent of each rule in this experimental set has only one event, we directly conceal the $\Delta t$ of every rule to make it become a TDR, and evaluate its precision on the test set. We use the following strategy for evaluation: given a TDR, we trade immediately according to the event saying in its consequent after its antecedent appears. Then we hold the position until the expected event happens or the maximum window size for consequent occurrence is reached. We can get the precision of a rule by computing the return in the process of the holdings.

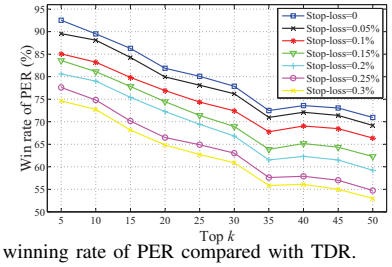



Fig. 5. The winning rate of PER compared with TDR.

For making a fair comparison between PER and TDR, we close out when the float loss exceeds a threshold during the holdings by a TDR. We tune such threshold and show the results in Figure 5. The results illustrate the average winning rate of PER at top $k$ under different stop-loss thresholds. We argue that PER has a better average precision when this measure is higher than $50\%$. From the figure, we can see that PER performs better than TDR under all the settings, though the winning rate decreases as $k$ increases. Here we vary the stop-loss threshold in a small range, i.e. $0$ to $0.3\%$, and the reason is that the daily price change ratio of each industry sector is small. We investigated the data and found $0.3\%$ is ranking at around 33th percentile among all ranked daily price change ratios of industry sectors. We can conclude from this comparison that with the constraints added by PER, precisions of rule could make valid contributions.

## V. Conclusion

In this paper, we formulate the problem of mining precise-positioning episode rules (MIPER), which is helpful for applications where automatic responses are needed in a timely manner. We propose one enumeration approach for MIPER and further devise two approaches based on a compact trie structure to enhance the pruning power as well as reduce execution time of the mining process. Experiments evaluate the efficiency of the proposed methods. Also, the effectiveness of precise-positioning episode rules is clearly demonstrated in a case study about China stock market.

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (No. 61602438, 91546122, 61573335, 61473273, 61473274), National High-tech R&D Program of China (863 Program) (No.2014AA015105), Guangdong provincial science and technology plan projects (No. 2015 B010109005).

## References

[1] Xiang Ao, Ping Luo, Chengkai Li, Fuzhen Zhuang, and Qing He. Online frequent episode mining. In *ICDE*, 2015.

[2] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, 2000.

[3] Heikki Mannila, Hannu Toivonen, and A Inkeri Verkamo. Discovery of frequent episodes in event sequences. *DMKD*, 1997.

[4] Cheng-Wei Wu, Yu-Feng Lin, S Yu Philip, and Vincent S Tseng. Mining high utility episodes in complex event sequences. In *KDD*, 2013.