Mining Episode Rules From Event Sequences Under Non-Overlapping Frequency

Oualid Ouarem[1], Farid Nouioua[1,2], and Philippe Fournier-Viger[3]

[1] Department of Computer Science, University of Bordj Bou Arréridj, Algeria
oualid.ouarem@univ-bba.dz,farid.nouioua@gmail.com
[2] LIS, UMR-CNRS 7020, Aix-Marseille University, France
[3] Harbin Institute of Technology (Shenzhen), Shenzhen, China
philfv8@yahoo.com

Abstract. Frequent episode mining is a popular framework for retrieving useful information from an event sequence. Many algorithms have been proposed to mine frequent episodes and to derive episode rules from them with respect to a given frequency function and its properties such as the anti-monotony. However, the interpretation of these rules is often difficult as their occurrences are allowed to overlap. To address this issue, this paper studies the novel problem of mining episode rules using non-overlapping occurrences of frequent episodes. The proposed rules have the form $\beta \Rightarrow \alpha$ where α and β are frequent episodes and β is a prefix of α. This kind of rules is well adapted for prediction tasks where a phenomenon is predicted from some observed event(s). An efficient algorithm named NONEPI (NON overlapping EPIsode rule miner) is presented and experiments have been performed to compare its performance with state-of-the-art algorithms.

Keywords: Frequent episode mining · Event sequence · Episode rules.

1 Introduction

Frequent Episode Mining (FEM) is an active subfield of data mining, which aims at retrieving important knowledge from temporal data [1–4, 13]. The input of that process is a single event sequence, which may encode various types of data such as user clicks of web sessions [10], alarms in a telecommunication network [13], and network intrusion data [16]. The objective of FEM is to find partially or totally ordered sets of events that occur frequently in an input sequence.

FEM algorithms have been designed to find frequent episodes using various frequency functions to count their occurrences (how many times each episode appears). The first type, called **dependent frequency functions**, consider that occurrences of an episode may overlap (share some common events). The main functions of this type are the *window-based frequency*, *minimal occurrence-based frequency* [13], *head frequency* [7], *total frequency* [7] and *non-interleaved frequency* [8]. The second type are the **independent frequency functions**, which do not allow episode occurrences to overlap. Some functions of this type are

the *non-overlapped occurrence-based frequency* [9] and the *distinct occurrence-based frequency* [12].

Since the seminal work of Mannila et al. in 1997 [13], many algorithms were proposed to efficiently extract all frequent episodes. The first algorithms are MINEPI and WINEPI [13]. WINEPI applies a level-wise search to find large episodes using a sliding window, and counts the number of windows that contains each target episode, while MINEPI searches for minimal windows containing each target episode. Then, episode rules can be derived from the obtained frequent episodes. A drawback of MINEPI and WINEPI is the duplicate counting of episode occurrences. To overcome this problem, Huang and Chang proposed two novel frequency functions called the head frequency and total frequency, and two algorithms[5]. The first one, called MINEPI+, adopts a depth-first strategy inspired by MINEPI while the second one, called EMMA, avoids the huge number of duplication and unnecessary duplicate checks of MINEPI using the head frequency. Unfortunately, under the head frequency function, an episode may have a higher frequency than its sub-episodes, which makes the result difficult to interpret. The total frequency function avoids this issue. That study also showed how to generate an episode rule using that frequency function.

Some other FEM algorithms use other types of constraints. For instance, 2PEM [11] and FCEMiner [6] find closed episodes. However, in general, dependent frequency functions suffer from over counting the number of episode occurrences to calculate an episode's frequency. Therefore, they consume too much time and memory space compared to independent frequency functions. Contrarily, an algorithm that relies on an independent frequency function can calculate the set of frequent episodes with minimal resources. The non-overlapped occurrence-based frequency is an independent frequency function used in many applications such as manufacturing [9].

To find strong relationships between events in a complex event sequence, FEM was extended to mine episode rules satisfying some minimum confidence constraint. Several algorithms where proposed [5] to find episode rules that first find frequent episodes using the head frequency to then generate rules from them. Algorithms such as WinMiner[14] and PPER [2] apply the minimal occurrence-based frequency to find episode rules with optimal window size. To the best of our knowledge, no studies deal with the problem of mining episode rules using the non-overlapped occurrence-based frequency function, although this frequency function can be argued to be more adequate for real applications due to not over counting occurrences. This paper addresses this research gap by proposing an efficient algorithm named NONEPI (NON overlapping EPIsode rule miner) for mining episode rules using non-overlapped occurrence-based frequency, by extending a depth-first search FEM algorithm proposed by Wan et al. [17].

The remaining of this paper in organized as follows. Section 2 explains preliminaries of FEM. Section 3 states the problem solved in this paper. The details of the proposed algorithm are given in Section 4. Experimental results are presented and discussed in Section 5. Finally, a conclusion is drawn in Section 6.

2 Frequent Episode and Episode Rule mining

This section briefly defines the main concepts used in episode mining and episode rule mining. An event sequence is defined as follows.

Definition 1 (Event sequence). . *Let E be a set of event types. An event is a pair (e_i, t_i) where $e_i \in E$ is an event type and t_i is an integer that represents its occurrence time. An event sequence S on event types E is an ordered set of events $S =< (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) >$ where t_1 and t_n are the starting time and ending time of S respectively, and for any integers i, j if $i < j$ then $t_i < t_j$.*

For instance, an example event sequence is presented in Fig. 1. This sequence contains 13 events on 6 different event types: $E = \{A, B, C, D, M, N\}$. It starts at time $t_1 = 52$ and ends at time $t_{13} = 67$.

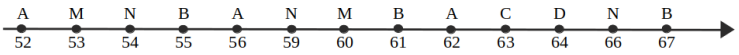

Fig. 1. An event sequence with 13 timestamps

The general definition of an episode is the following:

Definition 2 (Episode). *An episode α is a triple $(V, <_\alpha, g_\alpha)$ where V is a set of nodes $\{v_1, v_2, ..., v_n\}$, $<_\alpha$ is an order on V and $g_\alpha : V \rightarrow E$ is a mapping that associate each node with an event type.*

The integer n is the length of episode α. There are several particular types of episodes: when the order $<_\alpha$ is total, episode α is a *serial* episode. In this case, α is simply denoted as $\alpha = A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_n$ where each A_i is an event type. When the order $<_\alpha$ is trivial, episode α is called *parallel* and it is denoted as $\alpha = A_1 A_2 ... A_n$. Episode α is said to be **injective** if it does not contain any repeated event types, i.e., for any $1 \leq i, j \leq n$, if $i \neq j$ then $g(v_i) \neq v(v_j)$.

This paper focuses on discovering injective serial episodes. Thus, in the subsequent sections, any serial injective episode is simply called an episode. In the example sequence of Fig. 1, the event A is followed by event N, then followed by B, forming an episode with 3 events ($\alpha = A \rightarrow N \rightarrow B$) as they appear in that order in the sequence. The notion of sub-episode is formally defined as follows:

Definition 3 (Sub-episode). *Let $\alpha = A_1 \rightarrow \ldots \rightarrow A_n$ and $\beta = B_1 \rightarrow \ldots \rightarrow B_m$ be two episodes. β is said to be a **sub-episode** of α (denoted as $\beta \sqsubseteq \alpha$) if and only if: $\exists k, 1 \leq k \leq n - m + 1$ such that $\forall j, 1 \leq j \leq m$, $B_j = A_{k+j-1}$.*

In other words, β is a sub-episode of α if β is a part of α. Notice that if β is a sub-episode of α then it is easy to see that every occurrence of α contains an occurrence of β [13]. If $\beta \sqsubseteq \alpha$, we also say that α is a **super-episode** of β.

Two particular cases are distinguished: If $k = 1$ then β is located at the beginning of α and is called a **prefix** of α. In contrast, if $k = n - m + 1$ then β is situated at the end of α and is called a **suffix** of α.

Returning to the running example, the episode $\beta = A \rightarrow N$ is a sub-episode of the episode $\alpha = A \rightarrow N \rightarrow B$, denoted as $\beta \sqsubseteq \alpha$. Moreover, β is a *prefix* of α.

The notion of episode occurrence in a sequence and that of non-overlapped occurrences are formally defined as follows:

Definition 4 (Occurrence of an episode). *An occurrence of an episode α in an event sequence S is a vector $h = [t_1 t_2 ... t_n]$ where each t_i is an integer that represents the occurrence time (timestamp) of the i^{th} node of episode α.*

Given two occurrences $h = [t_1 t_2 ... t_n]$ and $h' = [t'_1 t'_2 ... t'_n]$, h and h' are said to be non-overlapped occurrences of episode α iff either $t_n < t'_1$ or $t'_n < t_1$.

In general, different sets of non-overlapped occurrences can be obtained for a given episode. We are in particular interested in maximal sets, i.e., containing a maximal number of non-overlapped occurrences:

Definition 5 (Maximal set of non-overlapped occurrences of an episode). *Given a set H of non-overlapped occurrences of an episode α, H is said to be a maximal set of non-overlapped occurrences iff for every other set H' of non-overlapped occurrences of α, it holds that: $|H| \geq |H'|$. We denote by $no(\alpha)$ the maximal set of non-overlapped occurrences of the episode α.*

For instance, consider the example sequence S of Fig. 1 and the episode $\alpha = A \rightarrow N \rightarrow B$. The set $H = \{[52\ 54\ 55], [56\ 59\ 61], [62\ 66\ 67]\}$ is the maximal set of non-overlapped occurrences of α in S, and hence $no(\alpha) = H$.

Definition 6 (Support of an episode). *The support of an episode α (denoted by $support(\alpha)$) is the cardinality of the maximal set of non-overlapped occurrences, i.e., $support(\alpha) = |no_\alpha|$.*

An episode α is frequent under non-overlapped occurrence-based support, if $support(\alpha) \geq minsup$ where $minsup$ is a user-defined support threshold.

In the previous example, we can see that the support of the episode α is 3.

To go further, episode rules captures binary relationships between couples of frequent episodes. An episode rule is an expression of the form $\beta \Rightarrow \alpha$ where α and β are two frequent episodes. An episode rule is said to be valid if the probability that its consequent occurs when its antecedent occurs is sufficiently high. This conditional probability associated to a rule defines the so-called confidence of this rule. It is given by : $conf(\beta \Rightarrow \alpha) = \frac{support(\beta \sqcup \alpha)}{support(\beta)}$ where $\beta \sqcup \alpha$ means the occurrence of β and α together. The exact interpretation of the $\sqcup$ operator depends on the kind of used rules and frequency. In our work, β is a sub-episode of α and hence $\beta \sqcup \alpha = \alpha$ (see Section 3). If the confidence of a rule is greater or equals to a user-defined confidence threshold $minconf$, then the rule is valid.

In general, different types of rules capturing different relationships between an antecedent and a consequence may be defined according to the application domain. This paper focuses on one particular type of episode rules which may be useful for prediction tasks.

3 Problem Definition

The proposed NONEPI approach uses the non-overlapped occurrence-based frequency to capture how often a rule is frequent. An episode rule is defined as follows:

Definition 7 (Episode rule). *An episode rule is an implication of the form: $\beta \Rightarrow \alpha$ where α and β are two frequent episodes (under non-overlapping frequency) and β is a prefix of α.*

The meaning of such episode rule is that if prefix β appears in the sequence, then its events or actions will strongly trigger all the remaining events necessary to form an occurrence of α. This may be helpful in understanding the root of a given phenomenon or to predict the future evolution of a set of ordered events.

Therefore, the confidence of an episode rule $\beta \Rightarrow \alpha$ is the ratio between $support(\alpha)$ and $support(\beta)$:

$$conf(\beta \Rightarrow \alpha) = \frac{support(\alpha)}{support(\beta)} \tag{1}$$

The main task of an episode rule mining algorithm is to efficiently find all valid episode rules at low cost (time and memory consumption). Thus, the problem to be resolved in this paper is formulated as follows: Given an event sequence S, a support threshold $minsup$ and a confidence threshold $minconf$, the task is to mine the set of valid episode rules, i.e., rules of the form $\beta \Rightarrow \alpha$ such that $conf(\beta \Rightarrow \alpha) \geq minconf$.

4 The Proposed Episode Rule Mining Approach

This section presents the proposed NONEPI approach to mine the set of frequent episode rules based on non-overlapped occurrences. The approach is divided into two steps: the first one consists in mining frequent episodes and the second one consists in mining the set of valid episode rules.

4.1 Extracting Frequent Episodes

The first step consists in mining frequent episodes according to their non-overlapped occurrences. This function (Algorithm 1) is inspired by the approach of Wan et al. [17], where the input is the minimum frequency threshold $minsup$.

To avoid considering all the search space, the frequent episode generation under the non-overlapped occurrence-based support count utilizes the following anti-monotony property.

Proposition 1. *Let α and β two episodes such that $\beta \sqsubseteq \alpha$, if episode α is frequent then episode β is also frequent. Equivalently, if episode β is infrequent then episode α is infrequent.*

Algorithm 1: Mining Frequent Episodes

Input: *minsup* - minimum support threshold
Output: F - the set of all frequent serial episodes

```
1  P= {}, F = {}, α = ∅
2  for each individual event e ∈ E do
3      no_e = {}
4  for k = 1 to n do
       % scan the sequence S
5      e = the event found at time t_k
6      no_e = no_e ∪ {[t_k, t_k]}
7  for each individual event e ∈ E do
8      if |no_e| ≥ minsup then
9          P = P ∪ {(e)}
10 F = P
11 for each individual episode α ∈ F do
12     for each individual episode β ∈ P do
13         no_{α⊔β} = Occurrence_Recognition(α, β)
14         if |no_{α⊔β}| ≥ minsup then
15             F = F ∪ {α ⊔ β}
16             α = α ⊔ β
17 return F
```

Proof. Since $\beta \sqsubseteq \alpha$ it follows that each occurrence of α in S includes an occurrence of β in S. The inverse is not necessarily true since it is possible to find an occurrence of β with a continuation which does not necessarily reach an occurrence of α. Hence, the number of occurrences of α is at most equal to the number of occurrences of β, i.e. $support(\alpha) \leq support(\beta)$.

Then if α is frequent, $support(\alpha) \geq minsup$. But since $support(\alpha) \leq support(\beta)$ it follows that : $support(\beta) \geq minsup$. We show in a similar manner that if β is infrequent then α is infrequent. ■

Algorithm 1 is applied as follows: First, it computes the frequent episodes of size one by scanning the event sequence S and for each event e occurring at time t_k in S, the time interval $[t_k\ ,\ t_k]$ is added to the set no_e of occurrences of e (lines 4-6). Then, the algorithm checks the support of these single event episodes to constitute the first frequent episodes (lines 7-9). After that, the function performs a depth-first search to find larger frequent episodes by successive calls of the *occurrence recognition* function which given an arbitrary episode α and an episode of size 1 β looks for the occurrences of $\alpha \sqcup \beta$ obtained by adding β to the end of α (lines 11-16). Notice that thanks to Proposition 1, the depth-first exploration continues only for nodes corresponding to frequent episodes (line 14). In other words, if an episode is not frequent, all the episodes that extend it will not be frequent and the corresponding search sub-space is pruned.

Algorithm 2 shows the details of the occurrence recognition function. This algorithm is inspired from [17] but thanks to the particularities of our context, the proposed algorithm avoids performing multiple sequence accesses to obtain

the set of occurrences of new episodes. The input is an episode to grow α and a single event episode β to grow α by. We get as output the occurrences of the new episode $\alpha \sqcup \beta$. For each occurrence of α, $O_i \in no_\alpha$, the algorithm browses the set no_β of occurrences of β to obtain an occurrence $O_j \in no_\beta$ which starts after the end O_i, i.e., $O_j.start > O_i.end$ where $O_j.start$ is the starting time of the j^{th} occurrence of β and $O_i.end$ is the ending time of the i^{th} occurrence of α. Then, we obtain a new occurrence $[O_i.start, O_j.end]$ of $\alpha \sqcup \beta$ (line(1-5)). To overcome the problem of overlapping occurrences, the function also removes any occurrences of α that overlaps with the new one of the new episode (line 6-7)).

Algorithm 2: Occurrence Recognition

Input: episode α - an episode to grow
episode β - a single event episode to grow α by.
Output: $no_{\alpha \sqcup \beta}$ - set of non-overlapped occurrences of new episode $\alpha \sqcup \beta$

```
1 for each O_i ∈ no_α do
2     for each O_j ∈ no_β do
3         if O_j.start > O_i.end then
4             update O_i.end = O_j.start
5             O_i.timestamps = O_i.timestamps ∪ O_j.start
6             for each O_k in no_α s.t. i < k ≤ m ∧ O_k.start < O_i.end do
7                 remove O_k from no_α
8             no_{α⊔β} = no_{α⊔β} ∪ O_i
9 return no_{α⊔β}
```

4.2 Extracting Episode Rules

The second phase of the process is to mine all valid episode rules of the form $\beta \Rightarrow \alpha$ where α and β are two frequent episodes under non-overlapping frequency and β is a prefix of α. Recall that valid rules are those having a confidence which is no less than a minimum confidence threshold $minconf$. The confidence of a rule is calculated according to Eq. 1. The consideration of all possible combinations of α and β to form an episode rule leads to a huge search space. To reduce this search space two techniques are used: (1) For a given episode α as a rule consequent, only its prefixes are candidates to be antecedents of that rule. This follows from the very definition of an episode rule; (2) For a rule consequent α, it can be shown that some prefixes may not be tested at all as antecedents of this rule because they surely lead to invalid rules. This is obtained by using the anti-monotony property at the rule level (see Proposition 2).

Proposition 2. *Let α and β be two frequent episodes such that β is a prefix of α. If the rule $\beta \Rightarrow \alpha$ is invalid then: (1) the rule $\beta' \Rightarrow \alpha$ is invalid for every episode β' which is a prefix of β ($\beta' \sqsubseteq \beta$), and (2) $\beta \Rightarrow \alpha'$ is invalid for every episode α' such that α is a prefix of α' ($\alpha \sqsubseteq \alpha'$).*

Proof. Let us prove the part (1) of the proposition. Let α and β be two frequent episodes such that $\beta \sqsubseteq \alpha$ and suppose that the rule $\beta \Rightarrow \alpha$ is invalid. Let $\beta' \sqsubseteq \beta$. From Proposition 1 we have: $support(\beta') \geq support(\beta)$. It follows that: $conf(\beta' \Rightarrow \alpha) = \frac{support(\alpha)}{support(\beta')} \leq conf(\beta \Rightarrow \alpha) = \frac{support(\alpha)}{support(\beta)}$. But, since $\beta \Rightarrow \alpha$ is invalid we have: $\frac{support(\alpha)}{support(\beta)} \leq minconf$. It follows that: $\frac{support(\alpha)}{support(\beta')} \leq minconf$, which means that $\beta' \Rightarrow \alpha$ is invalid.

The part (2) of the proposition can be proved in a similar way. ■

Based on Proposition 2, Algorithm 3 generates all valid episode rules from an input sequence S.

Algorithm 3: Extracting Episode rules

Input: S: sequence of events on E (event types set), $minsup$: support threshold, $minconf$: confidence threshold
Output: R: complete set of episode rules.

```
1  F = FrequentEpisodes(minsup)
2  R = ∅
3  for each episode α ∈ F do
4      stop = false
5      β = pred(α)/* returns the predecessor of episode α*/
6      while not stop and β ≠ NULL do
7          if |no_α|/|no_β| ≥ minconf then
8              R = R ∪ {β ⇒ α}
9              β = pred(β)
           else
10             stop = true
11 return R
```

The algorithm takes as input the sequence of events S on the event types E, the user-defined frequency threshold $minsup$ and confidence threshold $minconf$. The output is the complete set of valid episode rules of the form $\beta \Rightarrow \alpha$. Hereafter, for an episode α of length l, we denote by $pred(\alpha)$ (predecessor of α) the largest prefix of α, i.e., the prefix of α of length $l-1$.

Initially, the algorithm starts by extracting all frequent episodes with $minsup$ as a minimum support threshold (by a call of Algorithm 1). Then, for each frequent episode α the algorithm produces all the valid rules having α as consequent. Starting by $\beta = pred(\alpha)$ as a potential antecedent, the algorithm computes the confidence of $\beta \Rightarrow \alpha$. If this rule is valid it is added to the set of output rules and the algorithm passes to the next shorter prefix. The algorithm stops the exploration of prefixes of α as soon as it finds a prefix β such that $\beta \Rightarrow \alpha$ is invalid. Indeed, from Proposition 2, any further prefix β' will lead to an invalid rule too.

5 Experimental Results

As the proposed form of episode rules is introduced for the first time in this paper, there is no existing works about the same form that may be used for comparison with the performance of our algorithm. That is why the aim of this section is rather to understand some aspects of the behaviour of our algorithm. This section describes the experimental results obtained from two experiments carried out on synthetic sequences generated randomly. To randomly generate a sequence we specify : (1) its length, i.e., the number of events occurring in the sequence, (2) the number of event types and (3) the maximal duration between two consecutive events. We have used three types of sequences : large, medium and small. More information about the used sequences is given in Table 1.

Table 1. Information about used sequences

	Large	Medium	Short
1^{st} sequence	40 event types 500000 events	50 events types 100000 events	20 event types 29000 events
2^{nd} sequence	80 event types 600000 events	20 event types 130000 events	100 event types 350000 events
3^{d} sequence	100 event types 700000 events	100 event types 160000 events	50 event types 390000 events

5.1 Frequent Episode Mining

In this experiment, we have compared the proposed NONEPI algorithm with three well-known frequent episode mining algorithms that are implemented in Java and available in the SPMF library [15]: MINEPI which is a breadth-first algorithm using the minimal occurrence-based frequency as well as MINEPI+ and EMMA that are depth-first algorithms using the head frequency.

The comparison considers the variation, according to the support threshold, of (1) the number of found frequent episodes, (2) the maximal size of found frequent episodes as well as (3) the execution time. For each type of sequences we take the average value of the three used sequences (see Fig. 2).

Figures 2(a), (d) and (g) show the influence of *minsup* on the number of discovered frequent episodes in sequences of large, medium and short size, respectively. We can see that the number of generated frequent episodes in NONEPI is relatively more sensible to the variation of support threshold than in the other algorithms, where the number of generated frequent episode tends to remain stable for several successive values of *minsup*. We can also notice that for medium and large sequences, NONEPI generates less frequent episodes than the other

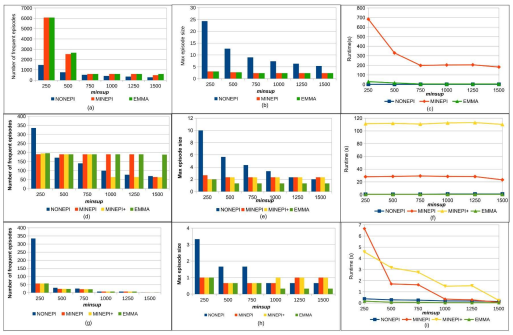

Fig. 2. Influence of *minsup* on the number of frequent episodes, the maximum episode size and the execution time for sequences of large, medium and short size.

algorithms. Indeed, as the size of a sequence increases, the episode occurrences that do not overlap tend to be less numerous than the overlapping occurrences.

Figures 2(b), (e) and (h) show the influence of the support threshold *minsup* on the maximal size of frequent episodes in sequences of large, medium and short size, respectively. We can see that as *minsup* increases, the maximum size of the discovered episodes by NONEPI decreases. However, that of MINEPI and MINEPI+ remains almost stable regardless of the value of support threshold. This means that the maximum size of frequent episodes is more sensible to support threshold in NONEPI than in the other algorithms. This is because NONEPI considers independent occurrences. Indeed, large episodes give rise to large and non overlapped occurrences that are naturally less frequent than overlapped occurrences or occurrences of small episodes. Thus, the large episodes that are frequent for small support thresholds become non frequent as soon as the support threshold increases.

Figures 2(c), (f) and (i) show the runtime comparison of the algorithms for different support thresholds in sequences of large, medium and short size, respectively. It is clear that the runtime of NONEPI and EMMA are significantly lower than that of the two other algorithms. MINEPI+ obtained the worst performance among all the tested algorithms for all types of sequences. For large sequences MINEPI+ fails to terminate before 800 seconds, that is why it is not represented in Figures (a), (b) and (c) of large sequences. The very good performance of NONEPI is explained by the fact that it scans the event sequence only once to extract timestamps of single event episodes and then joins them to obtain larger episodes occurrences while MINEPI and MINEPI+ scan the sequence more than once to find the occurrences of the newest episodes using the window width. This also shows the efficiency of the pruning strategy used in NONEPI and based on the anti-monotony property presented in Proposition 2.

5.2 Episode Rules

This experiment shows the proposed NONEPI for episode rule generation. It presents the variations of the execution time and the number of generated rules according to the minimum confidence threshold. We have represented the average values of the execution time and the number of generated rules, obtained in the nine tested sequences, and we have fixed the support threshold to $minsup = 100$ in order to focus on the confidence effect.

Fig. 3(a) and Fig. 3(b) show the influence of $minconf$ on execution time and the number of mined episode rules respectively. As the confidence threshold increases, the number of rules decreases slowly which means that generally, valid episode rules generated by NONEPI have high confidence values. By contrast, As the confidence threshold increases, the runtime decreases rapidly. This means that one may obtain the main valid episode rules even by using a relatively high value of confidence threshold which ensures a short runtime.

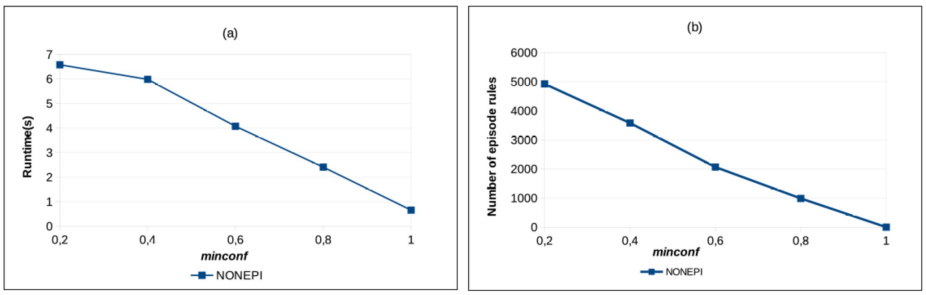

Fig. 3. Influence of $minconf$ on (a) execution time and (b) number of episode rules

6 Conclusion

This paper has proposed a new type of episode rules based on the non-overlapped occurrence-based support where an efficient algorithm, named NONEPI was presented. To the best of our knowledge, this is the only work that uses this support definition to mine all episode rules.

We have shown that our episode rules enjoy an anti-monotony property that allowed us to propose a pruning strategy that stops the generation of episode rules having a given episode α as consequent as soon as we find the largest prefix of α, say β such that $\beta \Rightarrow \alpha$ is invalid.

There are several possibilities of future work on NONEPI such as (1) extending it to process uncertain data using a probabilistic model or to mine episode rules with partially ordered events (2) considering more complex event sequences like streams and (3) building a sequence prediction model based on this algorithm.

References

1. Achar, A., Ibrahim, A., Sastry, P.S.: Pattern-growth based frequent serial episode discovery. Data and Knowledge Engineering, **87**, 91–108 (2013)
2. Ao, X., Luo, P., Wang, J., Zhuang, F., He, Q.: Mining Precise-Positioning Episode Rules from Event Sequences. IEEE Transactions on Knowledge and Data Engineering, **30**(3), 530–543 (2018)
3. Ao, X., Shi, H., Wang, J., Zuo, L., Li, H., He, Q.: Large-scale frequent episode mining from complex event sequences with hierarchies. ACM Transactions on Intelligent Systems and Technology (TIST, **10**(4), 1–26 (2019)
4. Fournier-Viger, P., Yang, Y., Yang, P., Lin, J.C.W., Yun, U.: TKE: Mining Top-K Frequent Episodes, In: Proc. 33rd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE'20), Springer (2020)
5. Huang, K.Y., Chang, C.H.: Efficient mining of frequent episodes from complex sequences. Information Systems. **33**(1), 96–114 (2008)
6. Huisheng, Z., Wang, P., Wang, W., Shi, B.: Discovering Frequent Closed Episodes from an event sequence, In: Proc. 2012 Int. Joint Conf. on Neural Networks (IJCNN'12), Brisbane, QLD (2012)
7. Iwanuma, K., Takano, Y., Nabeshima, H.: On anti-monotone frequency measures for extracting sequential patterns from a single very-long data sequence. In: Proc. 7th IEEE Conf. on Cybernetics and Intelligent Systems, pp. 213–217. IEEE (2004)
8. Laxman, S., Discovering frequent episodes: fast algorithms, connections with HMMs and generalizations, PhD thesis, Indian Institute of Science, Bangalore, 2006
9. Laxman, S., Sastry, P.S., Unnikrishnan, K.P.: Discovering frequent episodes and learning hidden markov models: A formal connection. IEEE Transactions on Knowledge and Data Engineering, **17**(11), 1505–1517 (2005)
10. Laxman, S., Sastry, P. S., Unnikrishnan, K.P.: A Fast Algorithm for Finding Frequent Episodes in Event Streams. In: Proc. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 410–419. ACM, New York (2007)
11. Liao, G., Yang, X., Xie, S., Yu, P.S., Wan, C.: Two-Phase Mining for Frequent Closed Episodes. In: Proc. 16th Int. Conf. on Web-Age Information Management. LNCS, vol 9658, pp. 55–66, Springer (2016)
12. Mahesh, J., Karypis, G., Kumar, V.: A Universal formulation of sequential patterns. Technical report 99-021, University of Minnesota (1999)
13. Mannila, H., Toivonen, H., Verkamo, I.: Dicovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery. **1**(3), 259–289 (1997)
14. Méger, N., Rigotti, C.: Constraint-based mining of episode rules and optimal window sizes, In: Proc. 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04), pp. 313–324 (2004)
15. SPMF Homepage, http://www.philippe-fournier-viger.com/spmf/ Last Accessed 01 Dec 2020
16. Su, M.-Y.:Discovery and Prevention of Attack Episodes by Frequent Episodes Mining and Finite State Machines. Journal of Network and Computer Applications. **2**(33), 156-–167 (2010)
17. Wan, L., Chen, L., Zhang, C.: Mining Dependent Frequent Serial Episodes from Uncertain Sequence Data, In: Proc. IEEE 13th Int. Conf. on Data Mining, pp. 1211–1216. IEEE (2013)
18. Zhu, H., Chen, L., Li, J., Zhou, A., Wang, P., Wang, W.:A General Depth-First-Search based Algorithm for Frequent Episode Discovery, In: Proc. 14th Int. Conf. on Natural Computation, Fuzzy Systems and Knowledge Discovery, pp. 890–899 (2018)