A Survey of Episode Mining

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
farid.nouioua@lis-lab.fr
[3] Shenzhen University, Shenzhen, China
philfv8@yahoo.com, philfv@szu.edu.cn

Abstract. Episode mining is a research area in data mining, where the aim is to discover interesting episodes, i.e., subsequences of events, in an event sequence. The most popular episode mining task is frequent episode mining (FEM), which consists of identifying episodes that appear frequently in an event sequence, but this task has also been extended in various ways. It was shown that episode mining can reveal insightful patterns for numerous applications such as web stream analysis, network fault management, and cybersecurity, and that episodes can be useful for prediction. Episode mining is an active research area, and there have been numerous advances in the field over the last 25 years. However, due to the rapid evolution of the pattern mining field, there is no prior study that summarizes and gives a detailed overview of this field. The contribution of this paper is to fill this gap by presenting an up-to-date survey that provides an introduction to episode mining and an overview of recent developments and research opportunities. This advanced review first gives an introduction to the field of episode mining and the first algorithms. Then, the main concepts used in these algorithms are explained. After that, several recent studies are reviewed that have addressed some limitations of these algorithms and proposed novel solutions to overcome them. Finally, the paper lists some possible extensions of the existing frameworks to mine more meaningful patterns and present some possible orientations for future work that may contribute to the evolution of the episode mining field.

Keywords: Temporal data mining, episode mining, pattern mining, sequential data analysis

1 Introduction

Data mining techniques are utilized to discover and represent implicit relationships present in large-scale data. These relationships can be made explicit for

users and can be manageable by different agents of intelligent systems. Various types of data can be analyzed by data mining algorithms, such as transactions, sequences, time series, and graphs. A type of data that is commonly found in real life is event sequences, which is an ordered list of events. In an event sequence, each event is associated with a time occurrence (for simplicity, we say occurrence). Such sequence may be, for example, a Web click stream, a sequence of points-of-interest visited by a tourist in a city, or a complex system's event log [67].

In some applications, it is desirable to perform sequential analysis to analyze a sequence database (a set of sequences). To address this issue, the task of sequential pattern mining (SPM) was proposed. SPM is a sub-field of data mining that aims at finding subsequences in data that reveal important relationships between data elements [23]. The support of a subsequence is the number of sequences where it appears. Finding frequent sequential patterns can be useful, for example, to find shopping patterns common to several customers (sequences) in a shopping dataset or sequences of words that appear in many sentences in a text. An important variant of SPM is sequential *rule mining*. The goal of sequential rule mining is to identify rules of the form $X \rightarrow Y$ that have a strong confidence (conditional probability) in a sequence database [64,26].

For other applications, it is required to analyze data that consist of a long sequence of events to discover interesting patterns or build models that can help understand the past or predict the future. To this end, the task of episode mining was proposed. Episode mining techniques are a set of techniques that can address this need. They are descriptive data mining techniques used to identify interesting relationships between events in a long event sequence. These relationships can take the form of *episodes*, which are subsequences of events that are deemed interesting according to some criteria such as their occurrence frequency, or implication-style rules, called *episode rules* that can reveal strong correlations between events. Episode mining has drawn the attention of many researchers and practitioners, in part because episodes provide information that can be interpreted by humans.

For example, an episode rule that may be found in the alarm log from a telecommunication network could indicate that it is common for an alarm A to be triggered in a machine after an alarm B occurs in another machine. Such rules provide information that can help understand the relationships between alarms and can be used to improve network maintenance by focusing on the most important alarms [61].

Episode mining [31,30] was utilized to extract patterns from sequential data in many domains such as to analyze telecommunication network data [30], discover patterns of exploit attacks to prevent network intrusions [56], analyze financial events and stock trends [3], analyze web logs, [67], detect intrusions

[55], identify internet worms [57], analyze the root-causes of machine faults [43], and detect network anomalies [54].

The first episode mining algorithms were introduced in 1997. Mannila et al. [30,31] defined the main episode mining task, which is to look for frequent episodes. Given a single long sequence of events, frequent episode mining (FEM) consists of finding the subsequences that appear frequently enough, i.e, whose number of occurrences in the input sequence is no less than a specified threshold called the minimum support (*minsup*). Those subsequences are called **frequent episodes**. For example, given a sequence of events from a server database's log, frequent episodes reveal sequences of service requests that are often repeated, which may then be optimized to improve performance.

Three main types of episodes have been extensively studied in prior work: serial episodes, parallel episodes [2,1,30,31], and injective episodes with an unrestricted partial order [5,17]. They respectively represent events that are totally ordered, appear at the same time, or are partially ordered. Besides these three basic episode types, various extensions of episode mining have been proposed. For instance, to reduce the number of episodes presented to users, subsets of episodes that have interesting properties have been studied, such as frequent closed episodes [8] and maximal episodes [84]. Moreover, additional constraints have been added to select more interesting episodes, such as constraints on the time gap between consecutive events [84]. Algorithms have also been developed to mine episodes in dynamic sequences, and to find the top-k most interesting episodes [44] instead of those whose frequency is not less than a given threshold. Another research direction that has been explored is to utilize alternative frequency definitions to identify frequent episodes such as window-based frequency [30,31], non-overlapped frequency [4,5,35,66], minimal occurrence-based frequency [39,21,31,85], head frequency [41] and total frequency [1,42] (see [12] for a review of different functions to measure frequency). Besides, some studies have also considered other criteria to select interesting episodes, such as their utility (importance), and evaluation functions that take into account uncertainty and/or imprecise information. In addition, many episode rule mining algorithms were published, which extend episode mining algorithms to derive implication-style rules, which can reveal dependencies between episodes [30,53,83,88]. Episode rules are suitable in practice for different tasks, namely the prediction of future phenomena or the explanation of some phenomena based on past events.

Episode mining is an active research area, and many advances have been made since the first paper, twenty-five years ago. However, we observe that there is no comprehensive survey on the topic. The contribution of this paper is to fill this gap by presenting an up-to-date survey that can serve both as an introduction to episode mining and as a guide to recent advances and research opportunities.

The rest of this paper is organized as follows: Section 2 introduces the basic concepts of frequent episode mining. Then, Section 4 presents the limitations of frequent episode mining and some extensions to address them. Section 5 discusses other pattern mining problems related to episode mining. Section **??** lists some lines of future research in the domain of episode mining. Finally, a conclusion is drawn in Section 7.

2 Frequent Episode Mining

Frequent episode mining is a popular framework to analyze temporal data [30,31]. It has been used in various domains where data contains time information. Frequent episode mining algorithms extract all frequent episodes either from a simple sequence of events [31,32,75] or a complex event sequence [15,22]. The former refers to a sequence where events are not allowed to occur simultaneously, while the latter allows simultaneous events.

2.1 Basic concepts

The typical input of an FEM algorithm is an event sequence and a user-defined integer threshold, called *minsup*, which represents the minimum number of occurrences that an episode must have to be considered as frequent.

Let there be a set of event types $E = \{E_1, E_2, ..., E_n\}$. An **event sequence** is a triplet $S = (s, T_s, T_e)$ indicating a list of events s that have been recorded between a time T_s and a time T_e. More precisely, the list s is an ordered set of event pairs of the form $s = \langle (I_1, T_1), (I_2, T_2), \dots (I_m, T_m) \rangle$ where $I_i \in E$ is an event type and T_i is the timestamp at which the event was observed ($1 \leq i \leq m$). Event pairs in s are ordered by increasing timestamps, and an event type may be observed multiple times in s.

For example, Figure 1 shows an example event sequence captured between time $T_s = 50$ and $T_e = 67$. This sequence may represent the event log of a database system, where event types $E = \{A, C, D, E, N, M\}$ may be disk operations such as to open, close, read, or write in a file. In that sequence, the following list of events has been recorded: $s = \langle (E, 52), (M, 54), (N, 55), (A, 56), ..., (A, 66) \rangle$. Note that in the following, the terms event and event type are used interchangeably when the context is clear.

An **episode** is a collection of events that appear together in a single sequence. Formally, an episode $\alpha = (V, \leq, g)$ is defined by a set V of nodes, a (partial or total) order $\leq$ over V and a mapping $g : V \rightarrow E$ that associates each node of V to an event type.

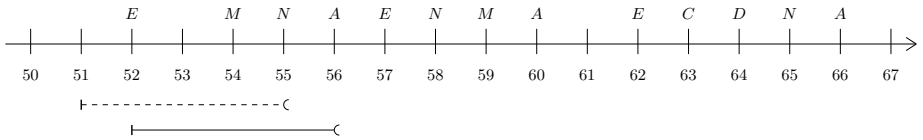
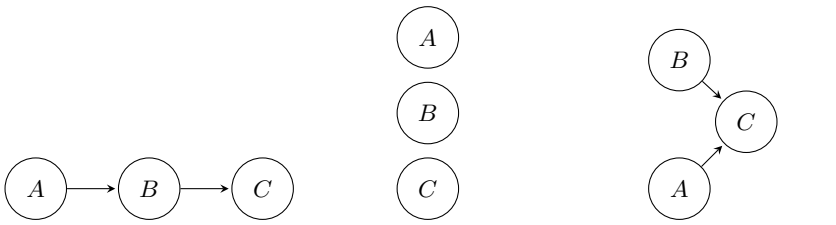


Fig. 1: An event sequence and two windows of length win=4

According to the provided order $\leq$, we may distinguish two particular types of episodes that are widely studied in the literature: If the order $\leq$ is total, the episode α is called a **serial episode**. In that case, an episode α can be denoted as $\alpha = A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_k$ such that $A_i \in E$ for all $1 \leq i \leq k$, and the notation α_i refers to the i^{th} event-type of episode α. And in the case where there is no constraint on the order between events, then α is called a **parallel episode** [11,87]. A parallel episode can be denoted as $\alpha = A_1 A_2 \ldots A_k$. It is also possible to find episodes that are a serial combination of parallel events, called **composite episodes** [84]. Moreover, the definition of episodes may be adapted to fit the nature of specific applications and data such as time-sensitive data and uncertain data [49,50]. These extensions will be discussed in subsequent sections.



(a) Serial episode (b) Parallel episode (c) Composite episode

Fig. 2: The three main episode types

For any episode $\alpha = (V, \leq, g)$, its size is denoted as $|\alpha|$ and defined as the number of events that it contains ($|\alpha| = |V|$). In the case where an episode α contains exactly k events, it is said to be a k-episode.

For instance, Figure 2 illustrates the three main types of episodes. Each episode displayed in that figure has a set of nodes $V = \{v_1, v_2, v_3\}$ and each node v_i is associated with an event-type as follows: $g(v_1) = A$, $g(v_2) = B$, and $g(v_3) = C$. Fig 2(a) shows a serial episode where the order is $\leq = \{(v_1, v_2), (v_2, v_3)\}$. This episode occurs in a sequence only if an event of type A occurs, is followed by an event of type B, and then by an event of type C. Fig 2(b) shows an example of a parallel episode without any order between its events ($\leq = \emptyset$). Finally, Fig. 2(c) depicts a composite episode where $\leq = \{(v_1, v_3), (v_2, v_3)\}$. This

episode indicates that events of types A and B must appear before C, but that there is no restriction on the order between events of types A and B.

An episode is said to be frequent in an event sequence if it occurs often enough in that sequence. Calculating the frequency of an episode requires selecting a frequency definition, which is how to count the number of occurrences of the episode in the event sequence. Let there be an event sequence S containing the list of events $s = \langle (I_1, T_1), (I_2, T_2), \ldots (I_m, T_m) \rangle$. An **occurrence** of an episode $\alpha = (V_\alpha, \leq_\alpha, g_\alpha)$ in S is a mapping $h : V \rightarrow \{1, 2, \ldots, m\}$ such that for all $v \in V_\alpha$ we have $g_\alpha(v) = I_{h(v)}$, and for all $w \in V_\alpha$ such that $v <_\alpha w$ we have $T_{h(v)} < T_{h(w)}$ [1].

For instance, consider the sequence depicted in Figure 1 and a serial episode $\alpha = E \rightarrow M$ with $V_\alpha = \{v_1, v_2\}$ and $g_\alpha(v_1) = E$, $g_\alpha(v_2) = M$. The events (E, 52),(M,54) are an occurrence of α because for $h(v_1) = 3$ and $h(v_2) = 5$ we have $g_\alpha(v_1) = I_{h(v_1)} = E$ and $g_\alpha(v_2) = I_{h(v_2)} = M$. Moreover, the order between events in the episode is preserved, i.e. since $v_1 <_\alpha v_2$, we have $T_{h(v_1)} = 52 < T_{h(v_2)} = 54$. It can be shown that the events (M,54),(E,62) are not an occurrence of α. Take a parallel episode $\beta = MN$ as another example. The events $(M, 58), (N, 55)$ and $(M, 58), (N, 59)$ are two occurrences of β.

The concepts of **sub-episode** and **super-episode** are important in episode mining. They refer to two types of dependency between pairs of episodes that have some common events (nodes). In general, given two episodes α and β, α is said to be a sub-episode of β (inversely, β is a super-episode of α) if and only if the nodes of α are a sub-set of the nodes of β and their order in α is compatible with that in β. This relationship is denoted by $\alpha \sqsubseteq \beta$. For example, a serial episode $\alpha = E \rightarrow N$ is a sub-episode of another episode $\beta = E \rightarrow M \rightarrow N$, and hence $\alpha \sqsubseteq \beta$. And a parallel episode $\gamma = EMN$ is a sub-episode of β but not of α.

It is to be noted that many algorithms use modified definitions of the aforementioned definitions to mine episodes that are more appropriate for a specific domain or to process specific types of data. For instance, the NONEPI algorithm (see [63]) relies on the concepts of suffix and prefix, which are special cases of the notion of sup-episode, and the 2PEM algorithm (see [28]) considers three types of sub-episodes called forward-extension, middle-extension, and backward-extension, in the context of closed episode mining, as it will be discussed later.

2.2 Frequency definitions in FEM

As mentioned before, the frequency of an episode (also called support) refers to how often an episode occurs in an event sequence. There exist multiple definitions of the frequency of an episode, which may lead to different calculations.

Generally, functions to calculate an episode's frequency can be categorized as either counting occurrences using a sliding window or not:

- **Windows-based frequency definitions.** Those definitions use a fixed-width window to capture the occurrences of any episode in the event sequence. The frequency of an episode is the number of windows of size k that contain at least one occurrence of the target episode. For instance, the windows-based frequency [30], head frequency [36], total frequency [36] and non-interleaved frequency [80] are examples of windows-based frequencies.
- **Occurrence-based frequency definitions.** Those are definitions that track exactly the occurrences of episodes in the entire event sequence without initially subdividing the sequence into sub-sequences except in cases where the user applies a constraint on the size of occurrences [2] called the *span* or *gap* constraints. This category of definitions includes minimal occurrence-based frequency [31], non-overlapped occurrence-based frequency [63,22] and distinct occurrence-based frequency [1].

Table 1 lists popular frequency definitions that belong to both categories and provides further details. For each definition, the type is given as well as some representative episode mining algorithm(s) that use that definition. Moreover, it is indicated if a definition satisfies the anti-monotony property. That property states that an episode must have a frequency that is no greater than those of its sub-episodes. This is a desirable property for designing efficient algorithms for frequent episode mining and will be discussed in the next subsection.

Table 1: An overview of episode frequency definitions

Type	Name	Anti-Monotonic	Representative algorithm(s)
Window-based	window-based frequency	Yes	WINEPI [30], EpiBF [14], WinMiner [53]
	head frequency	No	EMMA [41]
	total frequency	Yes	FEM-DFS [32]
	non-interleaved frequency	No	NOE-WinMiner [80]
Occurrence-based	minimal occurrence-based	No	MINEPI [30], MEELO [81], PartiteCD[76]
	non-overlapped occurrence-based	Yes	NONEPI[63], POERM [22]
	distinct occurrence-based	Yes	ONCE+ [48]

Window-based frequencies have some limitations. First, they can, in some cases, count multiple times the same episode occurrence if it appears in multiple windows. For instance, Figure 1 illustrates an example of an event sequence

with two time windows: $f_1 = (\langle(E, 52), (M, 54)\rangle, 51, 55)$ (the dashed line) and $f_2 = (\langle(E, 52), (M, 54), (N, 55)\rangle, 52, 56)$ (the plain line). Since these two windows contain E followed by M, a serial episode indicating that E is followed by M may be counted twice. Second, for a given window's width, an episode may be infrequent; however, the same episode may be frequent for another window's width. Thus, the frequency depends on the window's width, and the user may have to try various window widths to find an optimal one, which is time-consuming, or rely on other techniques to try to compute an optimal width. Fortunately, some recent algorithms, such as WinMiner, have proposed strategies to overcome that problem [53].

Most FEM algorithms are designed to discover episodes that occur frequently enough in a sequence under either a window-based or occurrence-based frequency. However, under an occurrence-based frequency, some long episodes may appear frequently but still contain some events that appear far from each other. And using a window-based frequency definition can ensure that events appear close together (in a window) but may miss episodes that are larger than the window size. Hence, other definitions have also been proposed. For instance, Cule et al. proposed an alternative definition and a depth-first search algorithm called DFS [18]. Cule et al. defined a set of constraints to capture the interestingness of a given episode, called the *cohesion* and *coverage*, defined as follows:

- *The coverage* measures how often an event of an episode appears in a sequence S and is calculated as:

$$P(X) = \frac{|N(\alpha)|}{|S|}$$

  where α is a given episode and $N(\alpha)$ is the set of all occurrences of its events, $P(\alpha)$ denotes the coverage of the episode $|\alpha|$ in the sequence S.
- *The cohesion* measures how close to each other are the events constituting an episode when appearing in a sequence. It is calculated as follows:

$$C(\alpha) = \frac{|\alpha|}{\overline{W}(\alpha)}$$

  where $\overline{W}(\alpha)$ is the average length of the shortest intervals (denoted as $W(\alpha, t)$) containing episode α, that is:

$$\overline{W}(\alpha) = \frac{\sum_{t \in N(\alpha)} W(\alpha, t)}{|N(\alpha)|}$$

  where

$$W(\alpha, t) = min\{t_2 - t_1 + 1 | t_1 \leq t \leq t_2 \text{ and } \forall e \in \alpha, \exists (e, t') \in S \text{ s.t. } t_1 \leq t' \leq t_2\}$$

Finally, the algorithm calculates the interestingness of an episode as follows:

$$I(\alpha) = C(\alpha) \times P(\alpha)$$

Consequently, given an interestingness threshold min_int, an episode α is said to be interesting iff $I(\alpha) \geq min\_int$. Using DFS, it was argued that the output contains coherent episodes that contain events close to each other [18]. Other definitions could also be devised for specific needs.

The next sub-section provides a discussion of two search space exploration strategies to identify frequent episodes that are the breadth-first and depth-first search. Then, an overview of the key techniques used by some of the most popular FEM algorithms is presented.

3 Breadth-first and Depth-first Strategies for Episode Mining

Since the definition of the problem of frequent episode mining by Mannila [30], many algorithms have been proposed to enhance the FEM process. These algorithms rely on different data structures and are designed to apply various frequency definitions. Generally, frequent episode mining algorithms can be classified based on their search strategy, which is either a breadth-first search or a depth-first search:

1. **Breadth-first algorithms.** These algorithms, also called level-wise algorithms, search for episodes by alternating between two steps called (1) candidate generation and (2) frequency check. The first step consists of generating candidate episodes from smaller episodes, while the second step is to count the frequency of those candidate episodes to determine if they are frequent. Breadth-first search techniques enumerate longer episodes and ignore many infrequent episodes by relying on the anti-monotonicity property. For instance, the CLOEPI [80] algorithm first scans the whole sequence of events to find the frequent 1-episodes. Then, CLOEPI re-scans the sequence from the beginning to identify frequent 2-episodes, 3-episodes, and up to m-episodes, where m represents the window's width. Some other level-wise algorithms that adopt a similar process are WINEPI, MINEPI [30], and FEM-BFS [2,80].
2. **Depth-first search algorithms.** These algorithms, such as MANEPI [35] and FCEMinner [34], scan the event sequence once to retrieve all 1-episodes and then try to recursively extend each episode by appending a frequent event while relying on the anti-monotonicity property for search space pruning. The depth-first strategy generally reduces the time and memory consumption for mining frequent episodes.

To design an efficient algorithm for FEM, the key issue is to avoid exploring the whole search space of episodes to retrieve the frequent episodes in the input sequence. This is important because the search space can be huge for relatively large sequences, which may lead to long runtimes and high memory requirements. To reduce the search space, a key technique is to use the anti-monotony property of the frequency measure. Let $sup(\alpha)$ denote the frequency of an episode α. The property states that for any two episodes α and β such that $\alpha \sqsubseteq \beta$ (α is a sub-episode of β), we have: $sup(\alpha) \geq sup(\beta)$. Consequently, if an episode α is not frequent, then all its super-episodes are infrequent as well and thus do not need to be explored which effectively reduces the time and memory consumption.

We next describe as an example the main process of two popular and representative FEM algorithms, namely WINEPI and FEM-DFS, which adopt a breadth-first and depth-first search, respectively.

WINEPI: A breadth-first search algorithm. WINdow-based **EPI**sode (WINEPI) is the first FEM algorithm to solve the problem of frequent episode discovery [30] and has been used in many other studies, such as to discover and prevent attack episodes [56] and to analyze mobile payment logs [51]. WINEPI takes as input a single long sequence of event S in the form already discussed in the previous sub-section, a set of event types E, a window width win and a user-defined minimum support threshold $minsup$ (also called minimum frequency threshold). The output of WINEPI is the set of all frequent episodes in the event sequence S with respect to the threshold $minsup$ and the sliding window size win. Like most breath-first search algorithms for pattern mining, WINEPI performs two main phases: (1) candidate generation and (2) frequency check:

- *Candidate generation.* It consists of generating candidate serial or parallel episodes (i.e: potentially frequent episodes). Figure 3 presents an example of a tree for episode enumeration. Initially, WINEPI builds a list to store episodes of size $l = 1$ (i.e: 1-*episodes*) by considering that each event type is an episode. Then, WINEPI performs a frequency check (we will discuss this step in detail after) to compute the set F_1 of frequent episodes of size $l = 1$ (i.e: 1-*episodes*). Then, WINEPI combines each pair of frequent episodes to obtain C_2, the set of candidates 2-episodes $\{< A, B >, < A, C >, < A, D > , < B, C >, < B, D >, < C, D >\}$. Then, in subsequent iterations, the 3-*episodes* are searched, followed by the 4-*episode* $\{< A, B, C, D >\}$, which is the largest candidate episode that can be obtained from these event types.
- *Frequency check.* This step consists of counting the frequency of candidate episodes using a sliding window over the event sequence. In WINEPI, the frequency of an episode α is the ratio between the number of windows w of length win that contain at least one occurrence of α and the total number of windows (denoted by $|W|$). Notice that there is a difference between tracking an occurrence of a serial and a parallel episode. To track occur-

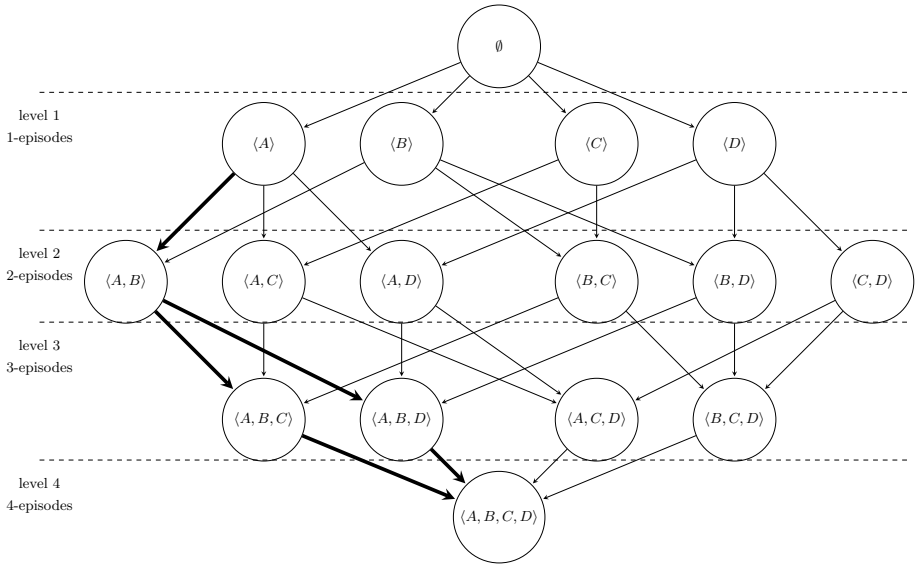


Fig. 3: A lattice representation of the breadth-first search carried out by WINEPI

rences of a serial episode, WINEPI and all algorithms that focus on serial episodes such as DiscoveryNonOver and DiscoveryTotal [2], UVSEM[1] (**U**nified **V**iew for **S**erial **E**pisodes **M**ining), and WINEPI [30] maintain a *Finite State Automaton* (FSA). The FSA that tracks all occurrences of an episode $\alpha = A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_k$ has $(k+1)$ states. The first k states are

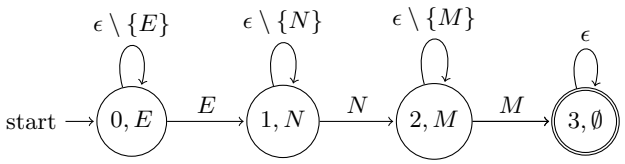


Fig. 4: Automaton that tracks all occurrences of $\alpha = E \rightarrow N \rightarrow M$

represented in the form (i, A_{i+1}) indicating that the automaton has reached the first i events and is waiting for the next event A_{i+1}. The last state $(k, \emptyset)$ is the accepting state, which means that, for a given window f, α occurs entirely in f and its frequency is incremented by one. Finally, WINEPI tests if the episode is frequent or not by comparing its frequency to *min_sup*. For instance, Figure 4 illustrates an automaton to recognize all occurrences of $\alpha = E \rightarrow N \rightarrow M$. Here, ϵ is the set of event types of the sequence S. For parallel episodes, as mentioned above, there is no constraint on the relative order of the episode's events. The single condition to take into consideration

is that all events in such an episode must occur inside a window to increment the episode's support.

WINEPI relies on the anti-monotonicity of the support (frequency) of episodes to reduce the search space, which has inspired many other studies. However, the window-based definition of support suffers from many problems. The main one is that of duplicate support counting; that is, for an episode α, the algorithm may count many windows for just one occurrence. For instance, consider the sequence given in Figure 1 with a window width $win = 4$. The serial episode $\alpha = E \rightarrow M$ is supported by 4 windows, while it can be seen that E is followed by M only twice. Hence, the resulting frequency does not describe the reality of events over the whole sequence. Therefore, an alternative to WINEPI was also proposed to solve this defect by considering the minimal occurrences [30] of the episodes. This alternative is called MINEPI (**MIN**imal occurrence-based **EPI**sodes mining). For more details on WINEPI and MINEPI and other frequency definitions, the reader may refer to [2,1,30,31,32,56]. The pseudo code of WINEPI is given in Algorithm 1.

Algorithm 1: The WINEPI algorithm

Input: E - the set of the event types,
min_sup - the minimum support threshold,
win - the sliding window width, S - an event sequence over E.
Output: F - the collection F of frequent episodes.

Scan the database to obtain the support of each event in the sequence S;
$F = \emptyset$;
$F_1 = \{e | e \in E \wedge sup(< e >) \geq min\_sup\}$;
$l = 2$;
while $F_l \neq \emptyset$ **do**
    $C_l = CandidateGeneration(F_{l-1})$;
    $F_l = \emptyset$;
    for *each* $\alpha \in C_l$ **do**
        if $FrequencyCheck(S, \alpha, win) \geq min\_sup$ **then**
            insert α into F_l ;
    $l = l + 1$;
    insert F_l into F;
return F

The WINEPI algorithm suffers from the following limitations: It cannot discover episodes with a number of events that is larger than the window size. To overcome that limitation, Casas-Garriga proposed a novel algorithm called EpiBF [14] that discovers unbounded episodes by automatically increasing the window length according to the episode length, using a parameter *tus* (time-unit separation). For each episode α, the algorithm calculates the frequency of α ac-

cording to its specific window of length *win* such that: $win = (\|\alpha\| - 1) \times tus$. The frequency of a given episode in a sequence s using a window width *win* is denoted as follows:

$$fr(\alpha, s, win) = \frac{|\{w \in W(s, win) | \alpha \ occurs \ in \ w\}|}{|W(s, win)|}$$

where $W(s, win)$ refers to the set of all these windows of size *win*.

FEM-DFS: A general depth-first search algorithm. Depth-first search algorithms such as FEM-DFS [32], *Extractor* [33], MANEPI [35], WinMiner [53] and POLYFREQDMD [75] first scan the event sequence to discover all frequent 1-*episodes*. To discuss those algorithms, it is useful to view the search space as an episode tree as displayed in Fig. 3, where each node represents an episode, the tree's root is the empty set, and child nodes of the root represent episodes of size 1. The key difference for DFS algorithms is that the tree is explored using a depth-first search. To enumerate larger episodes, a DFS algorithm forms a new episode (an extension) by concatenating a frequent episode (initially a 1-*episode*) with one of its siblings by considering the lexicographical order [2,1,34,68] (i.e $A \prec B \prec C \prec \cdots \prec Z$). This process is repeated recursively to build larger episodes (going down a tree branch) until no episode can be extended. Then, the algorithm backtracks to generate larger episodes from other nodes using the same process.

The depth-first search exploration process is illustrated with an example based on Fig. 3 (bold arrows). The DFS search starts by calculating the set of frequent episodes of size 1, $F = \{< A >, < B >, < C >, < D >\}$. The DFS strategy then grows each episode by combining it with 1-episodes (episodes of size 1) to make larger episodes. For example, the episode $< A >$ is combined with $< B >$ to obtain $< A, B >$. If that episode is frequent, the algorithm grows it with by combining it with the single episode $< C >$ to obtain a new episode $< A, B, C >$. If that episode is frequent it will be grown with $< D >$ to obtain $< A, B, C, D >$. If such an episode is infrequent, the algorithm backtracks on the same prefix and then try to combine it with another 1-episode. For example, given the episode $< A, B >$ and the 1-episode $< C >$, if the resulting episode $< A, B, C >$ is infrequent, the process keeps the episode $< A, B >$ and grows it with the 1-episode $< D >$. This process is pursued in the same way to explore the whole search space. Note that each new frequent episode is immediately added to the result set.

By using a DFS, the number of episodes that is kept in memory at any moment is generally much smaller than that of breath-first search algorithms, which reduces memory consumption. The combination process to generate episodes can also be more time-efficient.

Many algorithms give good results under a support definition. However, the mining process of these algorithms may need to be adapted if the support definition is changed and they may then generate more or less frequent episodes. For instance, an algorithm that uses the head frequency may not provide the same set of frequent episodes if the window frequency or any other frequency definition is used instead.

To provide a flexible algorithm supporting multiple definitions, Zhu et al. proposed a general algorithm called FEM-DFS [32] (Frequent Episode Mining in Depth-First-Search). It is an efficient algorithm for counting frequent serial episodes under an occurrence-based frequency definition called *earliest transiting* occurrences, which was proposed in [1]. Let h (as already defined in paragraph 2.1) be an occurrence of a **serial episode** $\alpha = A_1 \rightarrow A_2 \cdots \rightarrow A_k$. The occurrence h is said to be an *earliest transiting* occurrence of α if the first occurrence time $t_{h(A_i)}$ is the first occurrence of the event A_i after $T_{h(A_{i-1})}$ for $(2 \leq i \leq k)$. The set of all *earliest transiting* occurrences of α is denoted by $eto(\alpha)$. For instance, consider $\alpha = N \rightarrow A$. The events $(N, 55)(A, 56)$ constitute an earliest transiting occurrence of the episode α. The other two earliest transiting occurrences of α are $(N, 58), (A, 60)$ and $(N, 65), (A, 66)$.

The input parameters of FEM-DFS are an event sequence, a minimum support threshold, and a frequency definition *def_sup* to be used for calculating the support of episodes (i.e, total frequency, minimal occurrence-based frequency, etc.). FEM-DFS first extracts the set of all earliest transiting occurrences and then use them to calculate the support of any episode using the selected frequency definition. FEM-DFS explores the search space in a depth-first way to find larger episodes. After concatenating a pair of k-episodes, FEM-DFS computes the set *eto* of the resulting $(k+1)$-*episode*. Then, based on this set, FEM-DFS calculates the subset of the occurrences that satisfy the selected frequency definition *def_sup*. A similar process is used by the FEM-BFS algorithm (as called in [32]), which performs a breadth-first search but also utilizes the set of earliest transiting occurrence to obtain a global view of the event sequence for every collection of ordered events. The pseudocode of FEM-DFS is shown in Algorithm 2. As other depth-first search algorithms, FEM-DFS was shown to perform well in experiments [32].

In general, FEM algorithms differ from each other in: (1) the search strategy followed (depth-first or breadth-first search); (2) how they count the support to determine whether an episode is frequent enough or not; in other words, what kind of frequency definition is used in order to calculate the support of an episode; (3) what kind of episodes is considered to be calculated; and (4) the strategy for pruning the search space to reduce the amount of time and memory consumption. Moreover, there are other topics that attracted the interest of researchers, like the design of FEM algorithms that can perform well in particular practical contexts, such as in a cloud environment [7,8]. Moreover, the majority of existing algorithms consider the discovery of just one category of episodes,

Algorithm 2: General Algorithm of FEM-DFS

Input: E - set of event types
min_sup - minimum support threshold,
S - an event sequence over E,
def_sup: one of the support definitions.
Output: F-set of all frequent episodes

```
1  Scan the sequence to obtain P the set of frequent episodes of size 1.
2  F ← P
3  for each frequent episode α ∈ F do
4      for each frequent 1-episode β ∈ P do
5          if def_sup ≠' to' then
               // Grow the episode α with episode β
6              γ ← concat(α, β)
           else
               // Grow the episode β with episode α
7              γ ← concat(β, α)
           // The earliest transiting occurrences of concat(α, β)
8          eto(γ) ← ComputeETO(eto(α), eto(β), def_sup)
9          sup(γ) ← ComputeSup(eto(γ), def_sup)
10         if sup(γ) ≥ minsup then
11             F ← F ∪ {γ}
12 return F
```

which is that of sequential episodes, whereas many applications encounter too many events that happen simultaneously. Some recent works start considering this case study in many applications, such as medical applications [85]. This research line needs to be further explored. Besides, most of the algorithms that are designed for episode mining have only been applied in centralized systems. Hence, designing algorithms that perform on distributed systems or developing parallel algorithms for mining efficiently episodes constitutes a big challenge to increase the scalability and speed of the mining process.

4 Extensions of Traditional Episode mining

Although frequent episode mining has many applications, it can also be viewed as having limitations in terms of its assumptions. This section first discusses key limitations of FEM and then reviews some extensions that address these limitations. The identified limitations are:

– *A huge number of episodes.* In general, the minimum support threshold greatly affects the number of episodes that are discovered in an event sequence. For small threshold values, any FEM algorithm may discover a very large number of patterns, that can even be counted in millions. In those

cases, not only may algorithms have long runtimes and high memory usage, but analyzing the output can become very difficult for humans. Furthermore, the resulting set of frequent episodes may contain a lot of redundancy, since for example, if an episode is frequent, all its sub-episodes are also frequent for several frequency definitions. Fortunately, algorithms have been proposed for mining **concise representations** of episodes (see sub-section 4.1). The aim of these representations is to reduce the redundancy in the collection of frequent episodes without losing any information about them. Thus, these concise representations can be viewed as summarizing the whole set of frequent episodes. Algorithms for mining concise representations are generally faster than traditional algorithms.

- *Difficulty to set the minimum support threshold.* Another concern with traditional FEM algorithms is that it is difficult to find a suitable threshold value to obtain not too many episodes nor too few, but just enough. Without background knowledge, the user is left to find a good threshold value by trial and error, and a small change can give very different results. Thus, this process of fine-tuning the threshold can be very time-consuming. To address this issue, it was proposed to mine the **top-k frequent episodes** (see section 4.2), where k is set by the user and replaces the minimum support threshold.
- *The need for more constraints.* For time-sensitive applications, it is desirable to set additional constraints on the occurrences of episodes. For instance, some constraints may be set on the gap between two consecutive occurrences or on the duration of occurrences. An algorithm that allows constraints is said to be a *constraint-based episode mining algorithm*(see sub-section 4.3).
- *Frequency is not always the most important criterion to select patterns.* For some applications such as market basket analysis, alternative importance measures have been developed, such as the utility. It allows for assessing aspects such as the profit made by episodes instead of their frequency. A review of *high utility episode mining* is given in sub-section 4.5.
- *Unsuitable to mine episodes in a dynamic environment.* In some scenarios, it is desirable to mine episodes in a stream of events that is continuously updated rather than in a static sequence. Traditional FEM algorithms cannot update episodes incrementally. Many algorithms have been proposed to **extract frequent episodes from dynamic sequences** (see sub-section 4.4).
- *Events are considered equally important.* An assumption of traditional FEM is that all event types have the same importance. But in practical contexts, this is often not true. To address this issue, algorithms were proposed for weighted episode mining, where a weight is assigned to each event type (see sub-section 4.6).
- *Inability to deal with imperfect event sequences.* Real-life event sequences are often marred by imperfections. A prominent case is when occurrences of events are uncertain (see sub-section 4.7) or imprecise (see sub-section 4.8).

4.1 Episodes with concise representations

Many studies have been done on extracting concise representations of episodes to reduce the number of discovered episodes and have shown that those representations can provide better accuracy for various prediction tasks [7,8,28,33,34,45,48] [53,60,78,79,84].

In Table 2, we describe the characteristics of the main algorithms that use concise representations in terms of the frequency definition and the type of episodes targeted by each algorithm.

Table 2: Overview of algorithms with concise representations

Algorithm	Frequency definition	Episode type	Concise representation
LA-FEMH+[8]	minimal occurrence-based	Serial	Maximal Episodes
MaxFEM[24]	head frequency	Serial and Parallel	Maximal Episodes
Extractor[33]	minimal and non-overlapped occurrence-based	Serial	Generator Episodes
FCEMinner[34]	minimal and non-overlapped occurrence-based	Serial	Closed Episodes
2PEM[28]	minimal and non-overlapped occurrence-based	Serial	Closed Episodes
WFECM[4][45]	window-based frequency	Serial	Closed Episodes
MineEpisode[78]	minimal and non-overlapping occurrence-based	Serial and Parallel	Closed Episodes
CloEpi[79]	minimal occurrence-based	Serial	Closed Episodes
PPT/EPS[84]	minimal occurrence-based	Serial	Maximal Episodes

The most popular representations are maximal episodes [8], closed episodes [7,28,34,45,60,78,79], and generator episodes. Some of those representations, such as generator episodes, have been used to mine representative episode rules [33]. In the following, we formally define each of those episode types. The notation FE will be used to denote the full set of frequent episodes obtained from an input event sequence.

Closed episodes. Closed Episodes (CE) are frequent episodes that do not have any proper super-episode with the same support count, i.e.:

$$CE = \{\delta | \delta \in FE \wedge \nexists \delta' \in FE, \delta \sqsubset \delta' \wedge sup(\delta) = sup(\delta')\}$$

Algorithms such as CloEpi mine the set of frequent closed episodes by checking the forward, backward, or middle extensions of each frequent episode [79]. Put simply, this means to check if an event can be appended to a frequent episode before, inside, or after, to create a larger episode that has the same support. If yes, then the episode is closed, and otherwise, it is not.

Mining closed episodes instead of all frequent ones can greatly reduce the result set without losing any information. From the closed episodes, it is possible to recover all the frequent episodes without scanning the event sequence again. This property holds for frequency definitions that are anti-monotonic, and hence closed episodes are only used in this context. Some recent algorithms for mining frequent closed episodes include FCEMinner [34], CloEpi [79], MineEpisode [78] and 2PEM [28].

Maximal episodes. Maximal episodes (ME) [8,24,84] are the set of episodes that have no frequent super-episode, that is:

$$ME = \{\alpha | \alpha \in FE \wedge \nexists \beta \in FE \; s.t. \; \alpha \sqsubset \beta\}$$

An interesting property of maximal episodes is that $ME \subseteq CE \subseteq FE$. Thus, maximal episodes are a more compact representation than the closed episodes. There exist two algorithms that have been proposed for mining maximal episode mining in sequences called LA-FEMH+ [8] and [24]. The main purpose of these algorithms is to provide a smaller result set of episodes. In this context, an episode is viewed as *redundant* if it is a proper sub-episode of another frequent episode and it is not output. The LA-FEMH+ algorithm is restricted to discovering serial episodes, while MaxFEM can find both parallel and serial episodes.

Generator Episodes. Generator episodes (GE) are frequent episodes that have no sub-episode with the same support count and have closed super-episodes with the same support count [33], formally:

$$GE = \{\gamma | \gamma \in FE \wedge \forall \gamma' \sqsubset \gamma : \; sup(\gamma) \neq sup(\gamma') \wedge$$

$$\exists \delta \in CE : \; \gamma \sqsubset \delta \; \wedge \; sup(\gamma) = sup(\delta)\}$$

Contrary to the closed episodes, the generator episodes are not a lossless representation of frequent episodes. Zhu et al. proposed a depth-first strategy for stream prediction called *Extractor* [33] by computing frequent closed episodes with their generators and generating the set of all representative episode rules between episode generators and closed episodes. This algorithm mines the generator episodes following a depth-first search strategy and extracts the set of representative episode rules. It calculates the set of frequent episodes with respect to a support threshold under the minimal and non-overlapping occurrences-based

frequency. Then, it checks the closure of the discovered frequent episodes (see Subsection 4.1). Then, the algorithm calculates the generator episodes. Finally, the algorithm utilize the generator episodes and closed episodes to create the set of representative episode rules. The inputs of this step are the confidence threshold and the set of generator and closed episodes. *Given an episode rule γ, γ is said to be representative if there is no other episode rule γ' having the same support count and confidence, of which the antecedent is a subepisode of the antecedent of γ, and the consequent is a superepisode of the consequent of γ.* These rules can be viewed as meaningful for predictions as they attempt to predict the maximum amount of information (largest consequents) from the least amount of information (smallest consequents) for rules having the same support and confidence.

Some other types of frequent episodes have also been studied. Avinash et al. proposed the concept of injective episodes. An episode α is said to be injective if and only if all its event types are unique (there's no repetition of event types). The only algorithm that discovers such episodes is the one of Avinash et al. [5], which mines injective episodes with unrestricted order based on the non-overlapped frequency measure. Avinash et al. also proposed a new class of episodes that includes serial or parallel (injective or otherwise) episodes called *chain episodes* with an efficient algorithm to mine chain episodes [4]. There are few works that treat advanced types of patterns that could be mined from temporal data (closed, maximal, or generator episodes). Therefore, this is an excellent opportunity to advance the state of the art in episode mining with concise representations.

To reduce the number of discovered episodes, in addition to concise representations, Tatti introduced an additional measure called *episode significance* [59]. The computation of the significance of an episode is a post-processing step to test whether the discovered frequent episodes are truly interesting. The computation of the significance of such a frequent episode is based on minimal windows, such that a frequent episode is said to be *significant* if the average lengths of its minimal windows exceed the expected length according to the independence model.

4.2 Top-k frequent episode mining

Another extension of episode mining is top-k episode mining [27,44,71]. In this task, the user can directly indicate the number of episodes to be found. This task was proposed after observing that it is generally hard for users to determine an appropriate minimum threshold value. In fact, if the support threshold is too low, the mining process generates too many frequent episodes and hence requires a lot of memory or may have a long runtime. Similarly, if the threshold is set too high, the process may ignore many significant episodes. Running an algorithm

several times with different threshold values to find just enough episodes can be time-consuming.

Top-k episode mining algorithms provide a solution to this problem by letting the user directly set a parameter k that is the number of episodes k to be found. Then, a top-k algorithm will return the top-k most frequent episodes to the user. Hence, a benefit of those algorithms is that the user does not have to run an algorithm several times and adjust the minimum threshold to find a given number of episodes.

However, the main drawback of top-k episode mining algorithms is that this task is more difficult than traditional frequent episode mining if the respective parameters (k and the minimum threshold) are set to generate the same number of episodes [27]. This is because for top-k episode mining, no assumption can be initially made about the support of the top-k episodes to reduce the search space. Hence, top-k algorithms typically set an internal minimum support threshold to 0 and then start searching for episodes. Then, as soon as k episodes are found, the algorithms can increase the internal threshold and then continue searching. When no more episodes can be found, the top-k episodes are returned to the user. Due this search process, top-k episode mining have to explore a larger search space to find the same number of episodes as a traditional FEM algorithm [27].

4.3 Constraint-based episode mining

Several algorithms were designed for constraint-based episode mining. These algorithms integrate one or more constraints into the FEM process to filter out episodes deemed uninteresting. Constraints are user-specified criteria that frequent episodes must respect. Many kinds of constraints have been proposed in the literature. Constraints are either applied to the output of a FEM algorithm as a post-processing step (by discarding frequent episodes that do not meet the constraints) or during the search for frequent episodes. In terms of efficiency, it is preferable to integrate constraints in the search procedure for mining episodes, as some constraints can help reduce the search space. Constraint-based algorithms can be faster and consume less memory than traditional FEM algorithms, depending on the chosen constraints.

A representative constraint-based episode mining algorithm is *DiscoveryTotal* [2]. It introduces two types of constraints. The first constraint considers the maximum allowed time between the first and last events of an episode's occurrence (the so-called span constraint or expiry-time constraint). Using this constraint, the algorithm reduces the frequency of any episode by ignoring occurrences containing events that are too far apart. The second constraint is the maximum amount of time between two consecutive events (gap or inter-event constraint). The consideration of time constraints has been used in WinMiner [53], EPS, and PPT [84] among others. Researchers have also studied the integration of the

minimum and maximum number of events per episode (length constraint) for the analysis of heterochrony in developmental biology [65].

4.4 Frequent episode mining in a dynamic sequence

A limitation of traditional episode mining algorithms is the assumption that the input sequence is static. In fact, traditional episode mining algorithms are designed to be applied once to a single long event sequence to obtain relevant episodes. Then, if new events are logged or if the information about previous events is revised, the mining algorithms have to be run again from scratch to obtain up-to-date frequent episodes. Hence, the traditional task of FEM is not efficient in such situations. Fortunately, several online and stream episode mining algorithms have been designed [20,33,37,85,48,52,73,74,82,91,40,77]. Zhu et al. have proposed the *Extractor* algorithm for mining episodes in an event stream and derive prediction rules from them [33]. The proposed approach consists of discovering closed episodes and their generators in an event log, and then generating non-redundant rules (called representative rules) to be matched on the event stream. To also deal with the problem of mining episodes from event streams, Xing et al. proposed an algorithm called *MESELO* (**M**ining fr**E**quent **S**erial **E**pisodes via **L**ast **O**ccurrence) [82]. The new algorithm discovers episodes in an event stream by organizing events from the stream into smaller batches, such that at any time, the algorithm captures and stores only the latest incoming batch. The input of this algorithm is an event-growing sequence S, a support threshold *minsup*, a maximum windows size δ, and a window size Δ for the current valid sequence. The new problem of OFEM (**O**nline **F**requent **E**pisode **M**ining) consists in extracting all frequent episodes such that S contains the latest time stamps Δ, the size of each episode's occurrence is not greater than δ, and the frequency of the episodes is at least *minsup*. Algorithms such as ONCE and ONCE+ [48] were also designed for mining serial episodes with time constraints in a streaming sequence by searching for the last occurrence of each episode.

Some other extensions of episodes mining extend the event sequence representation in different ways to extract richer episodes. In the next sections, we summarize the most popular of these extensions.

4.5 High utility episode mining.

A limitation of traditional FEM is that it focuses on the frequency of events but treats all events as having the same importance. However, in real life, all event types are generally not equally important. High utility episode mining is a generalization of FEM where the utility (importance) of events is considered

to find important episodes [12]. Algorithms designed for *High Utility Episode Mining* (abbreviated as HUEM) consider as input a complex event sequence (where events may appear simultaneously) and where information is available about the external and internal utility of events. The external utility represents the relative importance of an event type in the whole sequence, while the internal utility represents the importance of an event when it occurred. All utility values are expressed as positive numbers, and the goal is to enumerate episodes having a utility that is no less than a minimum utility threshold (called high utility episodes).

HUEM is a difficult problem because the functions to calculate the utility are not anti-monotonic and thus cannot be directly used to reduce the search space. The first proposed algorithm, UP-Span [12], discovers all high utility episodes and their minimal occurrences. To reduce the search space, it relies on an upper bound on the utility that is anti-monotonic. Then, TSpan [29] and HUE-Span [25] were proposed as improved algorithms to further reduce the search space and achieve better performance on large databases.

An application of HUEM that has been studied in prior work is for stock investment prediction. A methology was presented combining HUEM and a genetic algorithm that provided better result than a compared method based on frequent episodes [87].

4.6 Weighted episode mining

Another interesting extension of FEM is weighted episode mining, which aims to extract episodes from multiple sequences. The concept of *weight* in this extension of FEM is utilized to measure the relative importance of each sequence. Weighted episode mining can be viewed as a type of utility-based episode mining framework, except that in the utility model, each event is associated with a weight. To the best of our knowledge, only *Liao et al.* [45] studied weighted episode mining. The problem definition is as follows: Let there be a collection D of sequences, a user-specified maximal window width $maxwin$, $minsup$ and a parameter $minsup\_wa$. The goal is to discover all weighted frequent closed episodes in D, where each episode $P =< e_1, e_2, ..., e_k >$ must satisfy the following conditions :
(i) $t_k - t_1 \leq maxwin$,
(ii) There is at least a sequence $S_i \in D$, such that $sup_i(P) \geq minsup$, and there is no any $P' \in S_i$, such that $P \subseteq P'$ and $sup_i(P') = sup_i(P)$,
(iii) $sup\_wa(P) \geq minsup\_wa$.
Here, $sup_i(P)$ is the support of the episode P in the i^{th} event sequence from D under window-based frequency and $sup\_wa(P)$ is the weighted support of P such that $sup\_wa(P) = \frac{N_I}{N} \times \sum_{i \in I}(\omega_i \times sup_i(P))$ where:
1) I is the set of sequences where P is a frequent closed episode.

2) N_I is the number of sequences that contain the episode P.
3) N represents the total number of sequences.
4) ω_i represents the weight of the i^{th} event sequence of D.

For instance, consider the collection of three sequences (S_1, S_2 and S_3 with the weights 1, 2 and 3, respectively) presented in Fig. 5(N=3). Let $P =< CDE >$ be an episode. P occurs respectively in S_1, S_2 and S_3 3, 3, and 4 times ($N_I = 3$). For $minsup\_wa = 10$, $minsup = 3$, and $maxwin = 5$, episode P is a frequent closed episode in all the sequences (under the support definition mentioned in [45]). Hence, its weighted support may be calculated as $sup\_wa(P) = \frac{3}{3} \times (1 \times 3+2\times 3+3\times 4) = 21$ and hence, P is a weighted frequent closed episode because $sup\_wa(P) \geq minsup\_wa$.

In general, weights in weighted-episode mining can be set using various methods depending on the application, and designing such methods could be further investigated. Some potential methods include setting them, by hand based on statistics, external data, or previous experience.

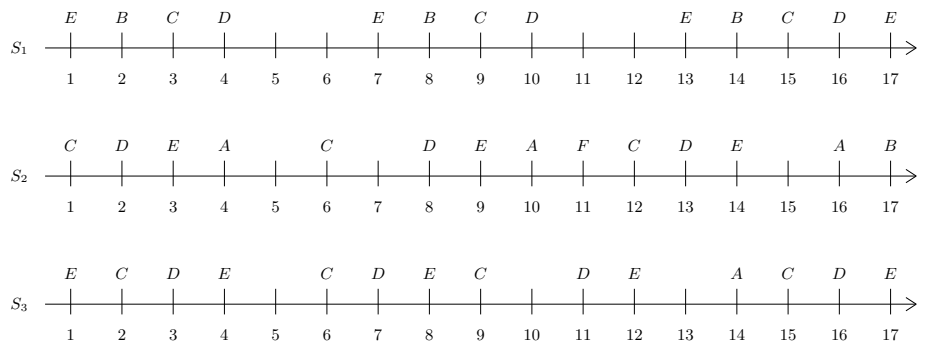


Fig. 5: A collection of three event sequences

Now, suppose that episode P occurs only in the sequence S_1 10 times ($N_I = 1$). In this case, the weighted support will be $sup\_wa(P) = \frac{1}{3} \times (1 \times 10) = 3.3$. Consequently, episode P will not be a weighted frequent closed episode.

4.7 Uncertain episode mining

Uncertain episode mining is an extension of episode mining that considers uncertain data. For many applications, logged events may be collected using noisy sensors, or the events themselves are inaccurate or imperfect, which makes traditional episode mining ineffective for their analysis. The main uncertainty model used in this new extension is the probabilistic episode mining model. According to this model, the user should set two thresholds: a minimum probability

threshold that indicates the confidence in the frequency of the episodes, and the support threshold to check whether generated episodes are probably frequent, i.e., the frequentness probability of a given episode must be no less than the probability threshold.

In the area of uncertain episode mining, the events are redefined such that each event is associated with a value p in the interval $[0, 1]$ indicating the probability that the event is present in the sequence. The resulting sequence of probabilistic events to be analyzed is then an uncertain sequence. Li et al. have formally defined these concepts and proposed some approaches to discovering probabilistic frequent serial episodes (P-FSE) [50], namely: (1) an exact approach that calculates the exact frequency of a given episode, (2) an approximate approach that approximates the frequency using a probability distribution of the frequency; and (3) an optimized approach that estimates an upper bound of the frequency without recognizing all episodes' occurrences. Another algorithm for uncertain episode mining is P-DFSE [49] for the discovery of probabilistic frequent serial episodes with respect to the dependence between the occurrences of a given episode. Those two algorithms calculate the frequency probability based on the *possible world semantics* theory such that, for a given episode, each occurrence is considered a possible world. An episode is considered a probabilistic frequent episode if and only if the calculated probability in a possible world [49] [50] is not less than the probability threshold.

4.8 Fuzzy episode mining

Fuzzy episode mining is a recent extension of traditional episode mining to deal with imprecise events using the fuzzy sets theory. This task considers sequences of events $S = \{E_1, E_2, \ldots, E_n\}$ and **event attributes** $A = \{a_1, a_2, \ldots, a_m\}$. Each event $E = \{E.a_1, E.a_2, \ldots, E.a_m\}$ consists of m values and an integer T that represents its occurrence in the sequence S. To the best of our knowledge, the only existing algorithm for mining fuzzy episodes was proposed by Luo et al. [39]. That work defines an **episode** $P(e_1, e_2, \ldots, e_k)$ as a set of **event variables**. An event variable of q attributes is denoted by $e^q\{attr_1 = v_1, \ldots, attr_q = v_q\}$ Because the attributes are *quantitative* (fuzzy) or *categorical*, a membership degree is associated with any value of an attribute a_i for a given fuzzy set or category of a_i. The occurrence of an episode is the product of the occurrences of its events variables in any event E. The proposed algorithm of Luo et al. uses the algorithm of Mannila et Toivonen that mines episodes under minimal occurrences [31].

Then, the problem is defined as follows: given an event sequence and a minimum occurrence threshold *min_occ*, the framework discovers all episodes with an occurrence frequency not less than *min_occ*. That work has demonstrated the applicability of fuzzy episode rules for real time intrusion detection.

5 Other Pattern Mining Problems related to episode Mining

The previous sections have described the problem of episode mining and some of its extensions. This section describes some other pattern mining problems that are strongly related to the episode mining problem.

Episode Rule Mining. The most important problem related to episode mining is that of Episode Rule Mining (ERM) in an event sequence. An episode rule is an implication $\beta \rightarrow \alpha$, where β and α are two frequent episodes. The interpretation is that if some episode β is observed, another episode α is likely to follow. The exact definitions of β, α and their dependence relation may take different forms, and their interpretation and practical use may vary from one algorithm to another. Finding episode rules can be useful to predict the future or to understand the data so as to perform recommendations or take useful decisions.

The concept of episode rules is similar to that of association rules [6] in traditional pattern mining, which consists of finding associations between sets of values (itemsets) that appear in a binary table. However, a key difference between association rules and episode rules is that the former do not consider time, while the latter are extracted from temporal data (a sequence of events). Thus, episode rules are useful to uncover temporal associations.

The process of extracting all episode rules is called episode rule mining. The first proposed approach to mine episode rules was proposed by Mannila et al.[30] in the paper where they introduced the WINEPI and MINEPI algorithms. The approach consists of first finding the frequent episodes and then generating episode rules by combining pairs of frequent episodes. To decide if a candidate rule is interesting, its confidence is calculated and compared with a confidence threshold. If the obtained value exceeds the minimum confidence min_conf, the concerned rule is output, and the user may utilize it to predict future events. Otherwise, the candidate rule is discarded. Numerous episode rule mining have been proposed [17,19,21,33,47,46,53,54,83,88,86].

The first designed algorithm MINEPI (mentioned above) adopts a breadth-first search for discovering episode rules. Among the algorithms that use a depth-first search, we note WinMiner [53] which mines efficiently all episode rules with a window constraint. Moreover, there exist some algorithms that perform the discovery of episode rules without candidate generation [47]. Various algorithms also have been proposed for analyzing event streams [19,33] and discovering utility-based episode rules from event sequences [88]. Episode rule mining has numerous applications, such as the analysis of telecommunication alarms [30], intrusion detection [55] and internet anomaly detection [54].

However, an episode rule requires a strict ordering between events. Thus, an episode rule mining algorithm may return many rules that are slightly different but describe the same situation in practice. To overcome this key problem, another kind of episodes is used to retrieve relationships between events, called partially-ordered episode rules. Recently, a new algorithm for mining partially-ordered episode rules called POERM [22] was proposed to find relationships between events in a complex event sequence. POERM efficiently finds all rules of the form $X \rightarrow Y$, where X and Y are two event sets. The problem tackled in POERM is the following: Given a complex event sequence S, three values $XSpan, YSpan, XYSpan \in \mathbb{Z}^+$, the minimum support and confidence thresholds $minsup$ and $minconf$, a rule $X \rightarrow Y$ is said to be valid iff: (1) the maximum duration of each occurrence of X and Y do not exceed some values $XSpan$ and $YSpan$ respectively; (2) the duration of the span between any pairs of occurrence of X and Y is at most equal to a value $XYSpan$ and (3) $sup(X) \geq minsup$ and $sup(Y) \geq minsup \times minconf$.

Some studies proposed other perspectives of how to compute the confidence of an episode rule. For instance, a collection of episode rule mining algorithms called MARBLES [17] was presented to mine all episode rules based on three new frequencies: (i) using *fixed windows*, *minimal windows* and *weighted minimal windows-based* frequencies. MARBLES also uses the *minimal-extensibility* and *weighted-extensibility* for minimal window-based frequency and weighted-frequency respectively, to find interesting episode rules. For such a frequency, a new confidence definition is used to capture episode rules that maximize the confidence.

While the majority of existing works discover only frequent episodes in simple or complex sequences, the main goal of knowledge discovery is to use this knowledge to detect dependencies between patterns to be used for explanation or prediction purposes. In the context of episode mining, these forms of associations are captured thanks to the notion of episode rules. Research in this direction aim is to develop new algorithms for extracting such rules in different contexts and interpreting them in practical applications.

Periodic episode mining. Periodic episode discovery [73] is a variation of episode mining. The goal is to find episodes that not only appear frequently in an event sequence or stream but that also occur periodically. To identify periodic patterns, the period lengths of each candidate episode are calculated, which is the elapsed amount of time between any two consecutive occurrences of the episode. An episode is said to be periodic if its period lengths are no greater than some threshold.

6 Some recent applications of episode mining

According to the previous discussions, it is apparent that frequent episode mining is an efficient framework to analyze sequential (temporal) data. Consequently, it has been used in many applications, and several recent studies have used it to enhance the performance of different systems. In spite of this, The majority of existing studies propose new algorithms based only on theoretical aspects, while building interactive tools for episode mining is an understudied topic. Fortunately, there exists an interactive tool called SNIPER [72] to analyze and understand the behavior of a recommender system using episode mining to enable the user to interact with the system to answer questions about the quality of recommendations. Various systems based on episode mining and tools can be built.

Frequent episode mining approaches have been extended to explore new ways of seeking important patterns in new practical domains like cyber-physical systems such as intelligent cars [90] or for monitoring in situ decommissioning sensor networks [89,9]. These recent works maintain a well-defined paradigm of prediction mechanism and mine frequent and abnormal episodes, respectively. For each framework, a specific FEM algorithm has been explicitly used to analyze the collected data and look for the sequential relationships between actions in each system. However, these two studies differ in the sequence construction step inside the framework. In the next subsections, we describe the process of those two frequent episode mining applications and show how the episode mining was used to fulfill the specified goal.

6.1 Event prediction in cyber-physical systems

A cyber-physical system (CPS) is a tight integration of cyberspace and physical components to achieve intelligent interactions with users. For instance, intelligent homes, smart cities, and autonomous cars are typical examples of CPS. These systems offer a large amount of data that can be used to build powerful tools with the aim of enhancing the performance of such systems or for prediction purposes.

Consequently, an efficient framework was proposed to predict the next event in a CPS like car usage, as in [90] where sequential events are analyzed in order to predict future events by using a graph data structure that enables building event chains (episodes). Mainly, there are three layers in the framework, and the fourth layer is composed of the application of the prediction model. In the fourth layer, we can integrate applications such as decision aid, system monitoring, and intelligent control. The three aforementioned layers are detailed as follows:

– **Event instance extraction.** It is the first layer of the framework. It consists of extracting event instances from incoming data from various physical components. Each component associates each event instance with its corresponding occurrence time. An event instance is composed of specific characteristics of any action, and it is represented by linked data as follows: $eventinstance =< URI, source, Attr >$ where URI is a unique identification of the event instance, $source$ is a link to a set of data from which the event instance is extracted and $Attr$ is a tuple (A, S, O, D, T, P) (A:action, S:subject, O: Object, D:Device, T:Time, P: place).

– **Event graph extraction.** After the extraction of event instances, sequential relationships between can be extracted between them. In the previous layer, we have shown that each event instance is associated with an occurrence time. The time field includes the starting and ending times. Consequently, all the event instances can be arranged in a single long event sequence, and thus, one can apply a FEM algorithm to extract several relationships between the events from the built sequence.
Formally, given a set of event instances $IS = [e_{i_1}, \ldots, e_{i_n}]$, the event sequence from IS is an ordered set $ES = \langle (e_1, e_1.str), \ldots, (e_n, e_n.str) \rangle$ where e_i is i^{th} event in the sequence and $e_i.str$ is the starting time of e_i. Next, an efficient FEM algorithm called MANEPI [35] is applied to discover the relationships from ES. Briefly, MANEPI is an efficient frequent episode mining algorithm from a long event sequence that uses a depth-first search under minimal and non-overlapping occurrence-based frequency. Besides, this work aims at mining frequent episodes by MANEPI while respecting a time gap constraint to reduce redundancy in the frequent episodes set. Since this approach uses a gap constraint, it is considered to be a constraint-based episode mining algorithm.
Finally, the framework builds the event graphs of each event chain (serial episodes) based on the sequential relations and the corresponding probability of transition between two adjacent events, such that:

$$P(e_{k_{i+1}}|e_{k_i}) = \frac{count(e_{k_i}, e_{k_{i+1}})}{count(e_{k_i})}$$

where $P(e_{k_{i+1}}|e_{k_i})$ is the conditional probability that describes the occurrence of $e_{k_{i+1}}$ knowing that e_{i_1} has already happened, $count(e_{k_i}, e_{k_{i+1}})$ is the total number of occurrences of $(e_{k_i}, e_{k_{i+1}})$ in all serial episodes.

– **Event prediction on the event graph:** It consists of building the prediction model from the event instance stream with the corresponding event graphs already extracted in the previous layer. To fulfill this goal, first, the framework identifies the event contexts and the candidate events. An event context refers to events that have occurred. A candidate event is a possible subsequent of such an event context. The second process in this layer is the prediction of the next events. A structured model called $SGNN$ of three stages was built to fulfill this process.

The three layers previously discussed constitute the important parts of the framework. There is another layer that constitutes the potential application. Consequently, the discussed work applied the proposed framework to car usage as Application layer. It can also be integrated into systems that depend on the IoT (Internet of Things) such as smart cities. For instance, the prediction mechanism could be used in various domains such as intelligent control, decision aid, behavior monitoring, and early warnings.

6.2 Frequent episodes from In-Situ Decommissioning sensor network

Another recent application that uses episode mining as an analysis technique consists of a set of sensors installed in a nuclear site called In-Situ Decommissioning (ISD). Episode mining techniques are used to process the large amount of data generated by ISD sensors [89,9] because the collected data is time-specific (each action is associated with a timestamp that constitutes its occurrence). ISD sensors are installed in the aim of monitoring the nuclear site. The collected data from the sensors at the nuclear site can be viewed as a single long sequence as defined previously. This data is composed of large records about (1) battery information, (2) strain information, (3) temperature information of four thermocouples, and (4) tilt-meter data (tilt degrees and local temperature of tilt-meter). For each sensor in the network, there exist two event types: High or Low. For example, given one of the four temperature sensors, say T1, if the captured value is higher than an adjacent data point, then the corresponding event type is T1H (high temperature). Otherwise, the corresponding event type is T1L (low temperature). Consequently, the events of four thermocouples constitute the set $[T1H, T1L, T2H, T2L, T3H, T3L, T4H, T4L]$. Similarly, the same concept is applied for the rest of the sensors such as for strain information $[SH, SL]$, for Biaxial tilt meter$[BH, BL]$, and for battery information (voltage) $[VH, VL]$. For the timestamps, the sensors also capture, for each action, its occurrence time.

Finally, the proposed approach uses the FEM algorithm presented in [68]. This algorithm uses the frequency definition based on non-overlapping occurrences. It is adapted for temporal analysis in the ISD sensor network in the aim of reducing the frequent episode searching time and improving the performance of the miner program. The final result of this application demonstrates the power of the FEM framework for analyzing temporal data. The approach also efficiently detects abnormal frequent episodes for ISD [9].

6.3 Intelligent collaboration framework for wheel manufacturing

In the last decades, the Internet of Things (IoT) had an increasing number of applications for many complex systems that are made of a large number of con-

nected devices, which themselves capture a large amount of data. Many complex systems interoperate with each other to perform a shared task, enhance the performance of such a process, or invoke new services by collaborating with services provided by different systems. The incorporation of services in IoT environments raises several challenges that must be overcome to fulfill its goals. Heterogeneity, changeable invocation logic, and spatio-temporal constraints are among the significant challenges of service collaboration. To overcome these challenges, Zhu et al. proposed a full framework of service collaboration for a wheel manufacturing process [58].

The proposed framework is very similar to the one proposed for CPS in [90]. Generally, the framework does five tasks. The first one is the event instance extraction from multi-source IoT data.

In this step, the framework models the event logic graph as linked data, where each ELG (Event Logic Graph) is denoted by: $ELG \leftarrow (URL, events, relations)$ such that: URL is a unique identifier of the ELG, events and relations are the set of events and the relations between them, respectively. Then, the framework constructs the event instances using a dispatcher, which feeds some detectors that recognize such an event. For example, the dispatcher may feed a detector that focuses on turning on a specific machine and another detector that increases its temperature. Notice that the detector can be a rule engine or a machine learning model.

Secondly, the framework constructs the event logic graph. In this step, the framework arranges the event instances into a sequence according to a chronological order. Then, an efficient episode mining framework is used to extract all significant episodes from sequential data. To do so, the framework follows an approach based on the well-known MANEPI algorithm [35] which is also used for Cyber-Physical Systems as mentioned above [90]. The inspired approach maintains a time gap constraint and a duration constraint to only recognize occurrences where both the time interval between two consecutive occurrences and the occurrences themselves are not too long. Hence, the set of event chains (serial episodes) does not contain any redundancy.

The final two tasks concern the application aspect. Their role is to predict future events. Among the practical applications where this prediction model may be used, we can cite, for instance, service recommendation, dynamic service composition, and service orchestration optimization.

The previous three application domains clearly show that FEM is an efficient tool to analyze temporal data. Other existing algorithms are potential tools for other applications. For instance, NonEpi [63] is an interesting algorithm for predicting diseases from observed symptoms since it discovers episode rules of the form $\alpha \rightarrow \beta$ such that α is the predecessor of β. A benefit of discovering

patterns that have the form of rules is that they can also be interpreted by humans.

6.4 Mining Train Delays

The railway network is a critical piece of infrastructure in any country. Delays may be very harmful and may cause shutdowns or serious disturbances to railway transportation. To build an intelligent solution to predict train delays [16], an episode mining algorithm called CloEpi [79] was used to reveal hidden patterns in sequential train data [92].

Before performing episode mining, data is preprocessed to obtain a single long sequence of events. The output of that step consists of a set of event sequences of delays of each characteristic point where the train passed, such that each event in such a sequence is denoted as *(Train_id, timestamp)* where $Train\_id \in N$ is a unique integer that represents the delayed train and *timestamp* is an integer representing the delay of the train at a given point. Finally, the algorithm CloEpi is used to analyze the sequence. This algorithm mines closed episodes to reduce the output size without any information loss.

7 Conclusion

Episode mining has been an active research field in the last few decades. It is used to analyze temporal data and help in understanding the behavior of systems, detect abnormalities, and predict the future.

We have presented in this paper an overview of that framework, and explained the concepts of episode mining and presented a classification of existing algorithms in terms of breadth-first and depth-first search. We have also discussed limitations of traditional episode mining algorithms and different extensions of traditional algorithms to deal with these limitations.

This paper is an up-to-date survey of the existing algorithms for discovering frequent episodes. It can serve both as an introduction to this domain and as a guide to recent research opportunities. The paper provides an overview of the different problems that can be solved by an episode mining framework in terms of criteria that each episode must respect. This can help in selecting an appropriate algorithm for a given application.

There are many opportunities for research on episode mining. Some of the research opportunities that have been mentioned in this survey are: (1) enhancing the performance of existing episode mining algorithms, (2) extending the serial

episode mining algorithms to consider parallel episodes, (3) extending episode mining to discover more complex or meaningful types of episodes, (4) designing algorithms to discover episode rules, and (5) building tools for episode mining. Besides that, researchers can draw inspiration from extensions to other pattern mining tasks such as sequential pattern [23] mining, sequential rule mining [64,26], itemset mining [93] and association rule mining [6], where many extensions have been proposed to address different needs but have not yet been used in episode mining.

References

1. A. Achar, S. Laxman, P. Sastry, A unified view of the apriori-based algorithms for frequent episode discovery, Knowledge and Information Systems, vol. 31, pp. 223–250, 2012.
2. A. Achar, I. A. Majeed, P. S. Sastry, Pattern-growth based frequent serial episode discovery, Data and Knowledge Engineering 87:91–108 (2013).
3. A. Ng and A. W.-C. Fu. Mining frequent episodes for relating financial events and stock trends. In Proceedings of The 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2003.
4. Achar, A., Sastry, P.S. Discovering frequent chain episodes. Knowledge and Information Systems, 60, 447-494.(2019)
5. Achar, A., Laxman, S., Viswanathan, R. et al. Discovering injective episodes with general partial orders, Data Min Knowl Disc (2012) 25: 67. https://doi.org/10.1007/s10618-011-0233-y
6. Agrawal, R. and Srikant, R., 1994, September. Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).
7. Amiri, M., Mohammad-Khanli, L. , Mirandola, R. Computing (2019), A new efficient approach for extracting the closed episodes for workload prediction in cloud https://doi.org/10.1007/s00607-019-00734-3
8. Ao, Xiang, Shi, Haoran, Wang, Jin, Li, Hongwei, He, Qing. (2019). Large-Scale Frequent Episode Mining from Complex Event Sequences with Hierarchies. ACM Transactions on Intelligent Systems and Technology.
9. Biswajit Biswal, Andrew Duncan, Zaijing Sun, ADA: Advanced data analytics methods for abnormal frequent episodes in the baseline data of ISD, Nuclear Engineering and Technology, Volume 54, Issue 11, 2022, Pages 3996-4004
10. B. Bouqata, C. D. Carothers, B. K. Szymanski, and M. J. Zaki, "Vogue: a novel variable order-gap state machine for modeling sequences," in PKDD, 2006.
11. Borgelt C., Picado-Muiño D. (2013) Finding Frequent Patterns in Parallel Point Processes. In: Tucker A., Höppner F., Siebes A., Swift S. (eds) Advances in Intelligent Data Analysis XII. IDA 2013. Lecture Notes in Computer Science, vol 8207. Springer, Berlin, Heidelberg
12. C.-Wei Wu, Yu-Feng Lin, P. S. Yu and Vincent S. Tseng, Mining high utility episodes in complex event sequences. Knowledge Discovery and Data Mining (KDD 2013).
13. Cappart Q., Aoga J.O.R., Schaus P. (2018) EpisodeSupport: A Global Constraint for Mining Frequent Patterns in a Long Sequence of Events. In: van Hoeve WJ.

(eds) Integration of Constraint Programming, Artificial Intelligence, and Operations Research. CPAIOR 2018. Lecture Notes in Computer Science, vol 10848. Springer, Cham
14. Casas-Garriga, G. (2003). Discovering Unbounded Episodes in Sequential Data. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds) Knowledge Discovery in Databases: PKDD 2003. PKDD 2003. Lecture Notes in Computer Science(), vol 2838. Springer, Berlin, Heidelberg.
15. Chen Y., Fournier-Viger P., Nouioua F., Wu Y. (2021) Mining Partially-Ordered Episode Rules with the Head Support. In: Golfarelli M., Wrembel R., Kotsis G., Tjoa A.M., Khalil I. (eds) Big Data Analytics and Knowledge Discovery. DaWaK 2021. Lecture Notes in Computer Science, vol 12925. Springer, Cham. https://doi.org/10.1007/978-3-030-86534-4_26
16. Cule, B., Goethals, B., Tassenoy, S., Verboven, S. (2011). Mining Train Delays. In: Gama, J., Bradley, E., Hollmén, J. (eds) Advances in Intelligent Data Analysis X. IDA 2011. Lecture Notes in Computer Science, vol 7014. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24800-9_13
17. Cule, B., Tatti, N., Goethals, B. (2014), MARBLES: Mining association rules buried in long event sequences. Statistical Analysis and Data Mining: The ASA Data Science Journal, 7: 93-110. https://doi.org/10.1002/sam.11199
18. Cule, B., & Goethals, B., Robardet, C. (2009). A New Constraint for Mining Sets in Sequences. Proceedings of the 2009 SIAM International Conference on Data Mining. 1. 317-328. 10.1137/1.9781611972795.28.
19. D. Chen and X. Zhang, "Internet Anomaly Detection with Weighted Fuzzy Matching over Frequent Episode Rules," 2008 International Conference on Apperceiving Computing and Intelligence Analysis, Chengdu, 2008, pp. 299-302. doi: 10.1109/ICACIA.2008.4770028.
20. D. Patnaik, S. Laxman, B. Chandramouli and N. Ramakrishnan, Efficient Episode Mining of Dynamic Event Streams, 2012 IEEE 12th International Conference on Data Mining, Brussels, 2012, pp. 605-614. doi: 10.1109/ICDM.2012.84
21. Dai H.K., Wang Z. (2013) Mining Serial-Episode Rules Using Minimal Occurrences with Gap Constraint. ICCSA 2013. Lecture Notes in Computer Science, vol 7971. Springer, Berlin, Heidelberg
22. Fournier-Viger P., Chen Y., Nouioua F., Lin J.CW. (2021) Mining Partially-Ordered Episode Rules in an Event Sequence. In: Nguyen N.T., Chittayasothorn S., Niyato D., Trawiński B. (eds) Intelligent Information and Database Systems. ACIIDS 2021. Lecture Notes in Computer Science, vol 12672. Springer, Cham. https://doi.org/10.1007/978-3-030-73280-6_1
23. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S. and Thomas, R., 2017. A survey of sequential pattern mining. Data Science and Pattern Recognition, 1(1), pp.54-77.
24. Fournier-Viger, P., Nawaz, M.S., He, Y., Wu, Y., Nouioua, F., Yun, U. 2022. MaxFEM: Mining Maximal Frequent Episodes in Complex Event Sequences. In: Surinta, O., Kam Fung Yuen, K. (eds) Multi-disciplinary Trends in Artificial Intelligence. MIWAI 2022. Lecture Notes in Computer Science(), vol 13651. Springer, Cham.
25. Fournier Viger, Philippe, Yang, Peng , Lin, Chun-Wei , Yun, Unil. (2019). HUE-Span:Fast High Utility Episode Mining. 10.1007/978-3-030-35231-812.
26. Fournier-Viger, P., Wu, C.W., Tseng, V.S., Cao, L. and Nkambou, R., 2015. Mining partially-ordered sequential rules common to multiple sequences. IEEE Transactions on Knowledge and Data Engineering, 27(8), pp.2203-2216.

27. Fournier-Viger, P., Yang, Y., Yang, P., Lin, J.C.W. and Yun, U., 2020, September. Tke: Mining top-k frequent episodes. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 832-845). Springer, Cham.
28. G. Liao, X. Yang, S. Xie and P. S. Yu,Two-Phase Mining for Frequent Closed Episodes, In WAIM 2016, Part I, LNCS 9658, pp. 55-66, 2016.
29. Guo G., Zhang L., Liu Q., Chen E., Zhu F., Guan C. (2014) High Utility Episode Mining Made Practical and Fast. In: ADMA 2014. Lecture Notes in Computer Science, vol 8933.
30. H. Mannila, H. Toivonen and A. I. Verkamo , Discovery of frequent episodes in event sequences, In Data Mining and Knowledge Discovery 1, 259-289, 1997.
31. H. Mannila, H. Toivonen, Discovering generalized episodes using minimal occurrence, In Proceedings ; KDD-96 .
32. H. Zhu, L. Chen, J. Li, A. Zhou, P. Wang and W. Wang, A General Depth-First-Search based Algorithm for Frequent Episode Discovery, In Preceedings of 14th IEEE International Conference on Natural Computation Fuzzy Systems and Knowledge Discovery, 2018
33. H. Zhu, P. Wang, W. Wang and B. Shi,Stream Prediction Using Representative Episode Rules, In Proceedings of the 11th IEEE International Conference on Data Mining Workshops,2011
34. H. Zhu, P. Wang, W. Wang, B. Shi, Discovering frequent closed episodes from an event sequence, In IJCNN 2012. IEEE, pp. 2292–2299.
35. H. Zhu, P. Wang, X. He, Yujia Li and Wei Wang, Baile Shi, Efficient Episode Mining with Minimal and Non-overlapping Occurrences, In Predeedings of IEEE International Conference on Data Mining, 2010
36. Iwanuma K., Takano Y., Nabeshima H. (2004) On anti-monotone frequency measures for extracting sequential patterns from a single very-long sequence. Proc. IEEE Conf. Cybernetics and Intelligent Systems, pages 213–217, Dec 2004.
37. J. C. C. Tseng, J. Gu, P. F. Wang, C. Chen, C. Li and V. S. Tseng, A scalable complex event analytical system with incremental episode mining over data streams, 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, 2016, pp. 648-655.
38. Jianxiong Luo, S. M. Bridges and R. B. Vaughn, "Fuzzy frequent episodes for real-time intrusion detection," 10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297),2001, pp. 368-371 vol.1. doi: 10.1109/FUZZ.2001.1007325
39. J. Luo and S. Bridges, M. Mining fuzzy association rules and fuzzy frequent episodes for intrusion detection. International Journal of Intelligent Systems,15(8):687–703, Jun 2000.
40. Jiawen Qin, Jinyan Wang, Qiyu Li, Shijian Fang, Xianxian Li, Lei Lei, Differentially private frequent episode mining over event streams, Engineering of Artificial Intelligence, Volume 110, 2022, 104681
41. K. Huang, C. Chang, Efficient mining of frequent episodes from complex sequences, Information Systems, vol. 33, no. 1, pp. 96–114, 2008.
42. K. Iwanuma, R. Ishihara, Y. Takano, H. Nabeshima, "Extracting frequent subsequences from a single long data sequence," In ICDM 2005. IEEE, pp. 186–193.
43. K. P. Unnikrishnan, Basel Q. Shadid, P. S. Sastry, and Srivatsan Laxman. Root cause diagnostics using temporal datamining. U.S.Patent no. 7509234, 24 Mar 2009.
44. Komate Amphawan, Julie Soulas, Philippe Lenca, Mining top-k Regular Episodes from Sensor Streams, Procedia Computer Science, Volume 69, 2015, Pages 76-85

45. Liao, Guoqiong, Yang, Xiaoting, Xie, Sihong, Yu, Philip, Wan, Changxuan. (2018). Mining Weighted Frequent Closed Episodes over Multiple Sequences. Tehnicki Vjesnik.
46. Lina Fahed, Armelle Brun, Anne Boyer, Influencer Events in Episode Rules: A Way to Impact the Occurrence of Events, Procedia Computer Science, Volume 60, 2015, Pages 527-536.
47. Lina Fahed and Armelle Brun and Anne Boyer, DEER: Distant and Essential Episode Rules for early prediction,Expert Systems with Applications,vol. 93,p.:283-298,(2018),https://doi.org/10.1016/j.eswa.2017.10.035
48. Li, Hui, Peng, Sizhe, Li, Jian, Li, Jingjing, Cui, Jiangtao, Ma, Jianfeng. (2018). ONCE and ONCE+: Counting the Frequency of Time-constrained Serial Episodes in a Streaming Sequence. Information Sciences. 10.1016/j.ins.2019.07.098.
49. Li Wan,Ling Chen and C. Zhang, Mining Dependent Frequent Serial Episodes from Uncertain Sequence Data, In Proceedings of IEEE 13th International Conference on Data Mining,2013
50. Li Wan , Chen, Ling and Zhang, Chengqi. (2013). Mining frequent serial episodes over uncertain sequence data. ACM International Conference Proceeding Series. 10.1145/2452376.2452403.
51. Li, X., Zhao, Y., Li, D., Guan, W. 2022. Error Serial Episodes Discovery from Mobile Payment Log in Distributed ETC. In: Lai, Y., Wang, T., Jiang, M., Xu, G., Liang, W., Castiglione, A. (eds) Algorithms and Architectures for Parallel Processing. ICA3PP 2021. Lecture Notes in Computer Science(), vol 13155. Springer, Cham.
52. M. Narmatha, Shri Hari Aravind .K, An Online Malicious Attack Detection using Honey Pot and Episode Mining, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 11, November 2016
53. Méger N., Rigotti C. Constraint-Based Mining of Episode Rules and Optimal Window Sizes. In PKDD 2004,pp 313-324
54. Min Qin and Kai Hwang, "Frequent episode rules for Internet anomaly detection," 3rd IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings., Cambridge, MA, 2004, pp. 161-168. doi: 10.1109/NCA.2004.1347773
55. Min-Feng Wang, Yen-Ching Wu, and Meng-Feng Tsai. Exploiting frequent episodes in weighted suffix tree to improve intrusion detection system. In Proc. Int'l Conf. Advanced Information Networking and Applications(AINA'08), pages 1246–1252, Mar 2008.
56. Ming-Yang Su, Discovery and prevention of attack episodes by frequent episodes mining and finite state machines, Journal of Network and Computer Application 33, pp. 156-167, 2010
57. Ming-Yang Su, Internet worms identification through serial episodes mining, ECTICON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Chiang Mai, 2010, pp. 132-136.
58. Min Zhu, Han Yu, Zhiyuan Liu, Bingqing Shen, Lihong Jiang, Hongming Cai, An intelligent collaboration framework of IoT applications based on event logic graph, Future Generation Computer Systems, Volume 137, 2022, Pages 31-41
59. N. Tatti, Significance of Episodes Based on Minimal Windows, in 2013 IEEE 13th International Conference on Data Mining, Miami, Florida, 2009 pp. 513-522.
60. Nikolaj Tatti, Boris Cule, Mining Closed Episodes with Simultaneous Events, Proceeding KDD '11 Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining Pages 1172-1180 (2011)

61. Nouioua, M., Fournier-Viger, P., He, G., Nouioua, F. and Min, Z., 2020. A Survey of Machine Learning for Network Fault Management. In: Machine Learning and Data Mining for Emerging Trends in Cyber Dynamics, Springer, Pages 1-27.
62. O. Quiroga, M. Joaquim and S. Herraiz (2012), Frequent and significant episodes in sequences of events: Computation of a new frequency measure based on individual occurrences of the events. In Proceedings of the International Conference on Know. Disc. and Info. Retrieval .p-p: 324-328.
63. Ouarem O., Nouioua F., Fournier-Viger P. (2021) Mining Episode Rules from Event Sequences Under Non-overlapping Frequency. In: Fujita H., Selamat A., Lin J.CW., Ali M. (eds) Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices. IEA/AIE 2021. Lecture Notes in Computer Science, vol 12798. Springer, Cham. https://doi.org/10.1007/978-3-030-79457-6_7
64. Pham, T.T., Luo, J., Hong, T.P. and Vo, B., 2014. An efficient method for mining non-redundant sequential rules using attributed prefix-trees. Engineering Applications of Artificial Intelligence, 32, pp.88-99.
65. R. Bathoorn, M. Welten, M. Richardson, A. Siebes, and F. J. Verbeek, Frequent Episode Mining to Support Pattern Analysis in Developmental Biology, (2010) In: Dijkstra T.M.H., Tsivtsivadze E., Lecture Notes in Computer Science, vol 6282. Springer, Berlin, Heidelberg
66. R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. In Proc. IEEE Int'l Conf. Data Mining (ICDM'03), pages 67–74, Nov 2003.
67. S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, A fast algorithm for finding frequent episodes in event streams. In KDD, 2007.
68. S. Laxman, P. Sastry, and K. Unnikrishnan, Discovering frequent episodes and learning hidden markov models: A formal connection, IEEE TKDE, 2005.
69. S. Laxman, P. Sastry, and K. Unnikrishnan, "Discovering frequent generalized episodes when events persist for different durations," IEEE TKDE, 2007.
70. S. Lin, J Qiao, An episode mining method based on episode matrix and frequent episode tree, Journal of Control and Decision, vol. 28, no. 3,pp. 339–344, **(2013)**.
71. Komate Amphawan, Julie Soulas, Philippe Lenca, Mining top-k Regular Episodes from Sensor Streams, Procedia Computer Science, Volume 69, 2015, Pages 76-85
72. Moens, S., Jeunen, O., Goethals, B. 2019. Interactive evaluation of recommender systems with SNIPER: an episode mining approach. In Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19). Association for Computing Machinery, New York, NY, USA, 538–539. https://doi.org/10.1145/3298689.3346965
73. Soulas J., Lenca P. (2015) Periodic Episode Discovery Over Event Streams. In: Pereira F., Machado P., Costa E., Cardoso A. (eds) Progress in Artificial Intelligence. EPIA 2015. Lecture Notes in Computer Science, vol 9273. Springer, Cham
74. Srivatsan Laxman, Vikram Tankasali, and Ryen W. White. Stream prediction using a generative model based on frequent episodes in event sequences. In Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD'09), pages 453–461, Jul 2008.
75. T. Katoh, H. Arimura, K. Hirata, An Efficient Depth-first Search Algorithm for Extracting Frequent Diamond Episodes from Event Sequences, IPSJ Online Transactions (2010)
76. T. Katoh, S.-I. Tago, T. Asai, H. Morikawa,J. Shigezumi, and H. Inakoshi, Mining Frequent Partite Episodes with Partwise Constraints, International Workshop on New Frontiers in Mining Complex Patterns,vol: 8399, pp:117-131,**(2013)**
77. T. You, Y. Li, B. Sun and C. Du, "Multi-Source Data Stream Online Frequent Episode Mining," in IEEE Access, vol. 8, pp. 107465-107478, 2020.

78. Tatti, N., Cule, B. Mining closed strict episodes, Data Min Knowl Disc (2012) 25: 34. https://doi.org/10.1007/s10618-011-0232-z
79. W. Zhou, H. Liu, and H. Cheng, Mining closed episodes from event sequences efficiently, In Proceedings of PAKDD 2010, pp. 310-318, 2010
80. Wu J., Wan L., Xu Z. (2012) Algorithms to Discover Complete Frequent Episodes in Sequences. In: Cao L., Huang J.Z., Bailey J., Koh Y.S., Luo J. (eds) New Frontiers in Applied Data Mining. PAKDD 2011. Lecture Notes in Computer Science, vol 7104.
81. Xiang Ao, Ping Luo, Chengkai Li, Fuzhen Zhuang, Qing He, Discovering and learning sensational episodes of news events, Information Systems, Volume 78, **2018**, Pages 68-80, ISSN 0306-4379.
82. X. Ao, P. Luo, C. Li, F. Zhuang, Q. He. Online Frequent Episode Mining, In Proceeding; ICDE Conference 2015
83. X. Ao, P. Luo, J. Wang, F. Zhuang and Q. He, "Mining Precise-Positioning Episode Rules from Event Sequences," 2017 IEEE 33rd International Conference on Data Engineering(ICDE),2017, pp. 83-86.
84. X. Ma,H. Pang, K.-L. TAN, Finding Constrained Frequent Episodes Using Minimal Occurrences. 4th IEEE International Conference on Data Mining: ICDM 2004 471-474.
85. Y. Cao, D. Patnaik , S. Ponce,J. Archuleta, P. Butler, W. Feng, N. Ramakrishnan, Parallel Mining of Neuronal Spike Streams on Graphics Processing Units. The Int J Parallel Prog (2012)
86. Y. Chen, P. Fournier-Viger, F. Nouioua and Y. Wu, "Sequence Prediction using Partially-Ordered Episode Rules," 2021 International Conference on Data Mining Workshops (ICDMW), 2021, pp. 574-580.
87. Y. Lin, C. Huang, V. Tseng, "A novel methodology for stock investment using high utility episode mining and genetic algorithm," Applied Soft Computing vol. 59, pp. 303–315, 2017.
88. Yu-Feng Lin, Cheng-Wei Wu, Chien-Feng Huang, Vincent S. Tseng, Discovering utility-based episode rules in complex event sequences, Expert Systems with Applications, Volume 42, Issue 12,2015
89. Z.J. Sun, A. Duncan, Y. Kim, K. Zeigler, Seeking frequent episodes in baseline data of In-Situ Decommissioning (ISD) Sensor Network Test Bed with temporal data mining tools, Progress in Nuclear Energy, Volume 125, (2020), 103372.
90. Z. Liu, H. Cai, H. Yu, B. Shen and L. Jiang, "Constructing the Sequential Event Graph for Event Prediction towards Cyber-Physical Systems," 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 1292-1297, doi: 10.1109/CSCWD49262.2021.9437885.
91. Zhongyi Hu, Wei Liu and Hongan Wang, Mining both frequent and rare episodes in multiple data streams, 2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Shenyang, 2013, pp. 753-761. doi: 10.1109/FSKD.2013.6816295
92. Infrabel Open Data portal, https://opendata.infrabel.be/pages/home Last Accessed 19 May 2023
93. J.M. Luna, P. Fournier-Viger, P. and S. Ventura, "Frequent itemset mining: A 25 years review," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 9, no. 6, e1329, 2019.