Discovering frequent parallel episodes in complex event sequences by counting distinct occurrences

Discovering Frequent Parallel Episodes in Complex Event Sequences by Counting Distinct Occurrences

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

[1]Faculty of Mathematics and Informatics, University Mohamed El Bachir El Ibrahimi of Bordj Bou Arreridj, El Anasser, 34030, Bordj Bou Arreridj, Algeria.
[2]LIS UMR-CNRS 7020, Aix-Marseille University, Marseille, France.
[3]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.

*Corresponding author(s). E-mail(s): philfv@szu.edu.cn;
Contributing authors: oualid.ouarem@univ-bba.dz;
farid.nouioua@gmail.com, farid.nouioua@lis-lab.fr;

Abstract

Event sequences are common types of data. Several episode mining algorithms have been developed to find episodes (subsequences of events) that appear frequently in an event sequence, with the aim of discovering useful knowledge for decision-making and predictions. However, most of these algorithms can only process simple event sequences (where, at most, one event occurs at each timestamp). In contrast, in many real-life applications, multiple events may occur at the same timestamp, resulting in complex event sequences. Moreover, numerous episode mining algorithms overestimate the frequency of episodes by counting the same events multiple times. As a solution, some algorithms have been designed to count only non-overlapping occurrences. Yet, it can be argued that this definition is too strict and discards many important events. To address these limitations, this paper presents an algorithm named EMDO (Episode Mining under Distinct Occurrences (EMDO) to find frequent episodes in a complex sequence by counting distinct occurrences. The proposed concept of distinct occurrences ensures that each event is not counted

more than once but allows distinct occurrences to overlap. A second algorithm, called EMDO-P, is also presented in this paper to derive strong episode rules in event sequences from episodes found by EMDO. To the best of our knowledge, this is the first study on mining frequent episodes using a frequency definition based on distinct occurrences. The experimental results confirm that the proposed algorithms are efficient.

Keywords: Complex Event Sequence, Frequent Episode, Episode Rules, Distinct Occurrence

1 Introduction

Data has become extremely valuable and plays an important role in daily life. They are captured in various ways such as through the Internet of Things (IoT) [29] and social networks [30]. They are used by decision makers, data analysts, and engineers to better understand various phenomena and provide improved services for real-world applications, such as product recommendation for e-commerce websites, analyzing energy usage in logistics or urban transportation, and even studying the human behavior in response to some specific events.

To incorporate the time dimension in data analysis, several temporal data-mining algorithms have been proposed in recent years. These algorithms can analyze a variety of temporal data such as time series and event sequences. An event sequence is a long sequence of events that are associated with time stamps.

To discover useful patterns in event sequences, Frequent Episode Mining (FEM) was introduced by Mannila et al. [8]. This framework has been applied successfully in many real-life applications, including alarm sequence analysis in telecommunication networks [8], user-behavior analysis from Web logs [26], and financial event and stock trend analysis [23]. The input data in the FEM is a single sequence of events, each of which is characterized by a type and an occurrence timestamp. The patterns discovered by an FEM algorithm are called *frequent episodes*, which are either subsequences or sets of events that occur frequently in an event sequence.

Since the seminal work of Mannila et al. [8], many recent studies have been conducted to improve the performance of FEM. In general, any FEM algorithm utilizes a frequency definition to find frequent episodes, which is based on a specific definition of episode occurrences. Generally, there are two types of frequency definitions: *dependent frequencies* that are based on occurrences which may share some events between them; and *independent frequencies* that only count occurrences that do not share any events between them.

In several studies, frequent episodes have also been used to derive a related pattern type called *episode rules* [8]. An episode rule is an implication that reveals a strong temporal relationship between two frequent episodes, and is

also observed in an event sequence. The interpretation of an episode rule is that if the left side of the rule occurs in the sequence, it will trigger the occurrence of the rule's right side shortly after with high confidence. Owing to the practical significance of mining episode rules in finding strong relationships, many algorithms have been proposed to mine these rules efficiently. Each algorithm uses a specific frequency definition and considers a given sequence type (either a simple or complex event sequence).

After reviewing the literature, two key limitations were identified. First, most FEM algorithms can only handle simple event sequences (where there can be at most one event per timestamp). However, it is not uncommon in real-world applications to encounter multiple events with the same timestamp, resulting in *complex* event sequences. Thus, traditional episode discovery algorithms cannot identify strong patterns with simultaneous events. Second, most frequency definitions count an event multiple times if there are overlapping occurrences, which may result in significant overestimation of the frequency of episodes. This is illustrated by the simple event sequence shown in Fig. 1, which contains three event types (a, b, c) observed at five timestamps (1,2, ...5).

A traditional episode mining algorithm considers that episode a *before* c appears four times at timestamps (1,4), (3,4), (1,5), and (3,5); thus, it has a frequency (support) of 4. However, this can be seen as an overestimation because each event is counted twice. For instance, event a at timestamp 1 is shared by occurrences (1,4) and (1,5). To address this problem, FEM algorithms have been designed to count only non-overlapping occurrences [10, 13, 19]. However, one could argue that this definition is too strict and may discard many important events, which may lead to an underestimation of the frequency of episodes. This is also illustrated by the sequence shown in Fig. 1. According to the non-overlapped occurrence-based frequency, episode a *before* c appears only once because a set of at most one non-overlapping occurrence can be found in that sequence, such as $\{(1,4)\}$ or $\{(2,4)\}$. However, this may seem unreasonable because there are two a that appear before two c in the sequence.

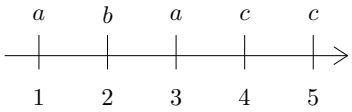


Fig. 1 A simple event sequence with five events and five timestamps

The extraction of distinct occurrences from complex event sequences has two major goals: first, to handle real-life phenomena with simultaneous events at a given timestamp. Hence, the resultant sequence will be a complex event sequence. This makes the analysis of such a sequence a challenging task that consumes a huge amount of resources. Second, the analysis should reveal hidden information without any duplicate occurrences of episodes or duplicate

combinations of events within the set of interesting patterns, since the purpose is to understand the targeted phenomena as much as possible. To achieve this, new techniques must be developed that allow simultaneous events to be taken into consideration in the calculation of the maximum possible number of episodes. As a consequence, we propose an approach to mining frequent parallel episodes in complex sequences under distinct occurrences. The proposed approach also enables the discovery of episodes that may occur at one timestamp since it does not consider any order between the nodes of such an episode.

To the best of our knowledge, no prior studies have attempted to jointly solve these two limitations. In this paper, we address them by proposing two new episode mining algorithms called EMDO (Episode Mining under Distinct Occurrences) and EMDO-P (EMDO with pruning strategy), based on a novel frequency definition that is less strict than the non-overlapped occurrence-based frequency, using a novel concept of distinct occurrences. The main intuition is to count the frequency of episodes by allowing overlapping occurrences, but not allowing the same event to be used twice to reduce the problem of underestimation. For example, in the sequence illustrated in Fig. 2, the episode *a before c* has a set of two distinct occurrences (1,4) and (3,5), since these occurrences do not reuse the same events, and thus the frequency of that episode is deemed to be two.

The two key contributions of this study are as follows: First, a novel frequency definition is defined for episodes and episode rules based on distinct occurrences of complex event sequences. Second, this definition is integrated into two novel algorithms, EMDO and EMDO-P, to efficiently find episodes and episode rules with this new definition, respectively. The experiments presented in this paper on various datasets show that the algorithms are efficient for different types of data and parameter settings.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents preliminaries and reviews the key concepts of frequent episodes and episode rules. Section 4 describes the studied episode discovery tasks and the novel algorithms in detail. Results from an experimental evaluation of the designed algorithms and a discussion is then presented in Section 5. Finally, Section 6 concludes the study and discusses future directions for research.

2 Related work

FEM is a data science task that has drawn the attention of many researchers, as evidenced by the increasing number of FEM algorithms. Since the initial study by Mannila et al. [8], several approaches have been proposed to enhance the efficiency of the FEM process by extending previous algorithms or mining new types of episodes or related patterns that reveal interesting relationships between events in a sequence.

Mannila et al. [8] introduced the first frequency definition, known as the *window-based frequency*. For a given episode, it counts the number of fixed-width windows in which the episode appeared at least once. Mannila et al. created an algorithm called WINEPI to mine such episodes. In the same paper, they also presented another algorithm called MINEPI, where the frequency is defined as the number of minimal windows that contain an occurrence of a target episode, and is called the *minimal occurrence-based frequency*. To overcome some of the limitations of previous approaches, two additional frequency definitions, namely, the *head frequency* [9] and *total frequency* [9], were also designed. For instance, Huang et al. [1] proposed two algorithms, EMMA and MINEPI+, to overcome the limitations of the window-based frequency using the head frequency [8].

Other studies have proposed other types of frequencies based on independent occurrences, such as the *non-overlapped occurrences-based frequency* [10] and the *distinct occurrences-based frequency* [11]. Another recent study proposed a frequency definition based on the *earliest transiting* occurrences [12], which provides a unified view of all previous frequency definitions. However the algorithm in [12] is limited to mining serial episodes with distinct occurrences using this unified view. This means that simultaneous events are not permitted, although they are common in many applications. An algorithm called ONCE + was also proposed to mine serial episodes in event streams [28].

Some studies have focused on identifying specific frequent episodes that met some additional conditions(s). In particular, several episode mining algorithms have been proposed for retrieving concise representations of frequent episodes. For instance, Xiang et al. [4] presented an algorithm called LA-FEMH+ that mines **maximal episodes**. An episode is maximal if and only if it has no proper frequent super-episode. Algorithms were also devised to discover **closed episodes** and **generator episodes**. A closed episode is a frequent episode with no proper super-episode and the same support (occurrence frequency). Closed episode mining algorithms include FCEMiner [14], 2PEM [15] under minimal and non-overlapping frequency, and Clo-episode [13] under minimal occurrence-based frequency. **Generator episodes** are frequent episodes that have No sub-episodes with the same support. The only algorithm that mines generator episodes is called Extractor [24]. Another area of research on episode mining is high utility pattern mining. The problem of mining high-utility episodes was defined as locating episodes with high importance, as measured by a utility function. The motivating application of this task is to identify episodes that are highly profitable. HUE-Span [17] and UP-Span [16] are efficient algorithms for mining utility episodes in event sequences.

Another recent variant of episode mining is the discovery of the top-k most frequent episodes in an event sequence, where k is a user-defined parameter. A modified version of the EMMA algorithm [1] called TKE was developed to perform this task [18]. Two other notable extensions of FEM are weighted episode mining [22] and fuzzy episode mining [25], which consider event sequences with varying weights. The former allows for the importance of different event

types to be weighted, whereas the latter handles events with quantities using fuzzy sets to deal with imprecise events. These two extensions can be viewed as forms of high utility pattern mining.

Some algorithms, such as PFSE [3] and D-PFSE [2] have also extended the FEM to model the data uncertainty of event sequences by using possible worlds semantics to mine frequent probabilistic episodes in uncertain sequences.

To extract strong relationships between sets of events in a complex event sequence, FEM was extended to mine episode rules that satisfy a minimum confidence constraint. Several algorithms have been proposed for this task. Generally, they discover the set of frequent episodes first and then evaluate the relationships between pairs of frequent episodes to build episode rules. Only episode rules with confidence above a user-defined confidence threshold are considered valid.

Mannila et al. proposed the first procedure for mining episode rules [8]. Further efficient algorithms have been proposed to handle time-sensitive applications such as program security trading using an algorithm called PPER [20]. These two algorithms use the minimal occurrence-based frequency.

In addition, the discovery of episode rules for event streams has been studied. For instance, the Extractor algorithm of Zhu et al. [24] finds closed episodes and their generators to identify nonredundant episode rules in an event stream under minimal and nonoverlapped occurrence-based frequency. Another technique, called MESELO, has also been proposed to mine episode rules in event streams too. The MESELO algorithm processes an event stream by decomposing it into smaller batches.

Recently, several algorithms have been proposed for mining partially ordered episode rules in a complex event sequence. For instance, POERM [6] uses the non-overlapped occurrence-based frequency and POERMH [7] is based on the head frequency. Furthermore, NONEPI [19] is an algorithm that performs a depth-first search to mine episode rules in simple event sequences using the non-overlapped occurrence-based frequency.

The analysis of existing works shows that most studies discover serial episodes in simple sequences, leaving many areas unexplored, such as other types of sequences, frequency definitions, and episodes. We focus on complex event sequences and parallel episodes with the distinct occurrence-based frequency definition, which is a very interesting topic in practice but has not been studied sufficiently. To the best of our knowledge, there is no algorithm that can mine parallel episodes and episode rules in a complex event sequence under a distinct occurrences-based frequency.

3 Preliminaries and problem definition

This section reviews the fundamental concepts used in Frequent Episode Mining (FEM) before giving a clear definition of the problem that we will be addressing in this paper: mining frequent episodes and valid episode rules in

a sequence under a frequency definition based on distinct occurrences. The input of FEM is an event sequence.

Definition 1 (Event) An event is a pair (e, t) where e is an element from a set E (set of all event types) that represents the event type and t is an integer that indicates the event's timestamp.

For instance, in TCP/IP network communication, an event occurring at a given time may be of type accept, representing the *accept* operation from a server receiving connections.

Definition 2 (Simple Event Sequence) Given a set E of event types, a simple sequence $S = \langle (e_1, t_1), (e_2, t_2), ..., (e_n, t_n) \rangle$ is an ordered set of events (e_i, t_i) such that $e_i \in E$ is the event type of the i^{th} event and t_i is its occurrence time in the sequence S. The sequence S is ordered, that is, for any integers i, j if $i < j$ then, $t_i < t_j$.

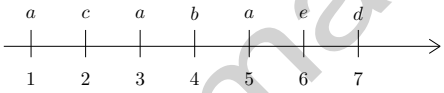


Fig. 2 A simple event sequence with 7 timestamps

For example, Fig. 2 provides a visual representation of a simple sequence with 7 events occurring at seven timestamps. The event types in this example are denoted by lower case letters, i.e. $E = \{a, b, c, d, e\}$. The formal definition of that sequence is $S = \langle (a, 1), (c, 2), (a, 3), (b, 4), (a, 5), (e, 6), (d, 7) \rangle$. For instance, this sequence may represent a list of web pages accessed through a web browser by a user, or its list of purchases in a web store.

A more general type of event sequence considered in this work is called a *complex sequence* and is simply referred to as *sequence* in the following.

Definition 3 (Complex Event sequence) Given a set E of event types, a complex sequence $S = \langle (\varepsilon_1, t_1), (\varepsilon_2, t_2), ..., (\varepsilon_n, t_n) \rangle$ is an ordered set of pairs (ε_i, t_i) such that $\varepsilon_i \subseteq E$ is a set of event types and t_i is the occurrence time of all events in ε_i in the sequence S.

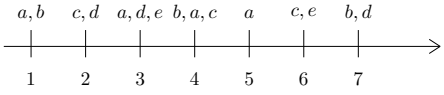


Fig. 3 A complex event sequence with 7 timestamps

An example of a complex sequence is shown in Fig. 3. This sequence contains 7 event sets and the same event types as in the previous example: $E = \{a, b, c, d, e\}$. The sequence starts at time $t_1 = 1$ and ends at time $t_7 = 7$. The sequence illustrated in the previous figure may represent the logs of a server in which the events are logged simultaneously. We can observe that the server logs, for each timestamp, different types of events that represent an action such as, for example, a request to establish a channel between a client and a server on a specific port. Here, the same port may be targeted by several machines. Therefore, the server should log the incoming requests for a given port at each timestamp, which clearly form a complex event sequence.

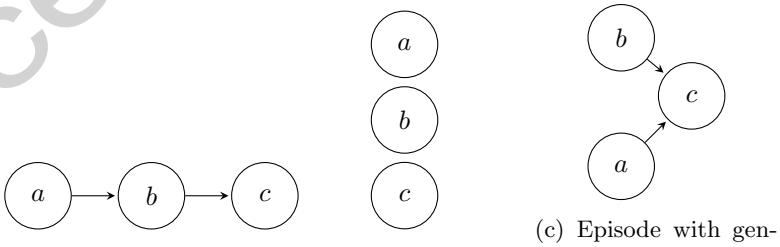
The goal of FEM is to discover patterns called *episodes*. A general definition is as follows:

Definition 4 (Episode) An episode α is a triple $(V, <_\alpha, g_\alpha)$ where V is a set of nodes $\{v_1, v_2, ..., v_n\}$, $<_\alpha$ is an order on V and $g_\alpha : V \rightarrow E$ is a mapping that associates an event type to each node.

According to the nature of the order $<_\alpha$, we can distinguish between different kinds of episodes:

- if the order $<_\alpha$ is total, α is a **serial** episode. In this case, α is denoted by $\alpha = A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_n$ where each A_i is an event type. In this case, the events must occur in the exact order specified by the episode. (see Figure 4.a for an example)
- If the order is trivial, the episode α is called **parallel** episode and it is denoted as $\alpha = A_1 A_2 ... A_n$. In this case, the events may occur in any order. An example is shown in Figure 4. b.
- Another type of episodes has been studied, called episode with general partial order [5–7]. In this case, event occurrences are partially ordered. Figure 4.c shows an example of such an episode.

In addition, an episode α is said to be **injective** if it does not contain any repeated event types, that is, for any $1 \leq i, j \leq n$, if $i \neq j$ then $g(v_i) \neq v(v_j)$.



(a) Serial episode (b) Parallel episode (c) Episode with general partial order

Fig. 4 The three main episode types

In this study, we focus on discovering parallel injective episodes. Hence, in the following sections, we use the term episode to refer to any injective parallel episode.

For instance, establishing TCP/IP communication in a client/server architecture forms a serial episode. Here, a TCP connection is a sequence of actions, such as *request()*, *accept()*, *send()*, and *receive()*. First, the client sends a *request* message requesting a connection to a specific port on the server. Then, the server must *accept* the request sent by the client to allow the sending and receiving of data. Note that there is a strict order between these events (TCP operations) to establish a link; hence, it is a typical example of a serial episode. In contrast, user clicks on a website may constitute a parallel episode because the user's behavior may not depend on the order between the clicks on the website.

In the example sequence, shown in Fig. 3, the events a, b and c can form an episode in that sequence, and this episode is denoted by $\alpha = abc$. Additional definitions are introduced to formally define how an episode α occurs in a sequence.

Definition 5 (Sub-episode) Let $\alpha = A_1 \dots A_n$ and $\beta = B_1 \dots B_m$ be two episodes. β is said to be a **sub-episode** of α (denoted as $\beta \sqsubseteq \alpha$) if and only if there exist m integers $i_1, i_2, \dots, i_m$ such that : $1 \leq i_1 < \dots < i_m \leq n$ and $B_1 = A_{i_1}, B_2 = A_{i_2} \dots B_m = A_{i_m}$.

In other words, β is a subepisode of α if the events of β are a subset of those of α. Note that if β is a subepisode of α then it is easy to see that every occurrence of α contains an occurrence of β. [8].

For instance, consider episode $\alpha = abc$ from the sequence of Fig 3. By definition 5, episode $\beta = ac$ is a sub-episode of $\alpha = abc$ ($\beta \sqsubseteq \alpha$).

The task of FEM consists of identifying all the frequent episodes in an event sequence. To achieve this, it is first necessary to select a frequency definition to count the occurrence of an episode. Generally, the number of occurrences of an episode is called its support, and an episode is frequent if and only if its support is not less than a user-specified threshold, *minsup*. The notion of an episode occurrence is defined as follows:

Definition 6 (Occurrence of episode, Distinct occurrences) Let S be a sequence and $\alpha = A_1 A_2 \dots A_n$ be an episode.

- An occurrence of the episode α in the sequence S is a vector of integers $h = [t_1 t_2 \dots t_n]$ such that each t_i is the occurrence time (timestamp) of the i^{th} node (event) of the episode α, i.e., A_i occurs at time t_i in S.
- Given two occurrences $h = [t_1 t_2 \dots t_n], h' = [t'_1 t'_2 \dots t'_n]$, h and h are said to be distinct if and only if $t_i \neq t'_j$ for all $t_i \in h$ and $t'_j \in h'$ and $1 \leq i \leq \|\alpha\|$ and $1 \leq j \leq \|\alpha\|$.

Then, the maximal (w.r.t. set inclusion) set of distinct occurrences of an episode is defined as:

Definition 7 (Maximal set of distinct occurrences) Let H be a set of distinct occurrences of episode α: H is said to be maximal if and only if for every set H' of distinct occurrences, $\|H\| \geq \|H'\|$. The notation $do(\alpha)$ denotes the maximum set of distinct occurrences of α.

For instance, consider the example sequence in Fig. 3. The maximal set of distinct occurrences of episode $\alpha = abc$ in that sequence is $do(\alpha) = \{[1\ 1\ 2],[3\ 4\ 4],[5\ 7\ 6]\}$. FEM algorithms have been designed to mine frequent episodes using various frequency definitions, each capturing a notion of how often an episode occurs in an input sequence. Under the distinct occurrences-based frequency, the support of an episode merely corresponds to the maximum number of its distinct occurrences in the sequence.

Definition 8 (Support of episode, Frequent episode) Let S be a sequence and α be an episode:

- The support of α under the distinct occurrences-based frequency definition (denoted by support(α)) is the cardinality of the maximal set of its distinct occurrences, that is $support(\alpha) = \|do(\alpha)\|$.
- An episode α is frequent under distinct occurrence-based frequency if and only if $support(\alpha) \leq minsup$ i.e., $support(\alpha) = \|do(\alpha)\|$, where $minsup$ is a user-specified threshold.

Frequent episodes are interesting because they can capture frequent relationships between events. To find patterns that are more actionable, several studies have focused on discovering episodes in the form of rules called *episode rules* [8, 19]. The concept of the episode rule is similar to that of association rule used in traditional frequent itemset mining [31]. Generally, an episode rule is an expression of the form $\alpha \Rightarrow \beta$ where α and β are two frequent episodes. This represents a binary relationship between two frequent episodes according to a specific frequency definition. To evaluate such a rule, the confidence measure, which is the probability that the consequent of an episode rule appears when its antecedent is observed, is commonly used. An episode rule is said to be *valid* if and only if its confidence exceeds the confidence threshold, denoted by $minconf$. Note that the exact definition of the confidence of an episode rule may vary depending on the algorithm and type of episode rules that are extracted.

The approach presented in this paper is called EMDO, which stands for **E**pisode **M**ining under **D**istinct **O**ccurrences. It relies on distinct occurrences-based frequency to capture how often a rule is frequent. An episode rule is then formally defined as follows:

Definition 9 (Episode Rule) An episode rule is an implication of the form $\alpha \Rightarrow \beta$ where α and β are two frequent episodes under the distinct occurrence-based frequency.

As mentioned above, episode rules reveal important relationships between frequent episodes. The meaning of an episode rule under Definition 9 is that if an episode α appears in sequence S, it will trigger the occurrence of episode β. Since we are using parallel episodes, any event of α can trigger the episode β. To capture the idea that β is a consequence of α we require that the beginning (resp. the end) of β must be after the beginning (resp. the end) of α. Therefore, the new form of episode rules studied in this paper covers many other works as in [6, 7] that mines partially ordered episode rules as well as episode rules under non overlapped occurrence-based frequency [19].

Definition 10 (Episode Rule Occurrence) Consider an episode rule $\alpha \Rightarrow \beta$ and two occurrences α_i and β_j of episodes α and β respectively ($\alpha_i \in do(\alpha)$ *and* $\beta_i \in do(\beta)$). An occurrence of the rule $\alpha \Rightarrow \beta$ is a vector $h = [t_{\alpha_1}...t_{\alpha_n} t_{\beta_1}....t_{\beta_m}]$. An occurrence h is said to be a **valid occurrence of the rule** if and only if: $T_s(\alpha) < T_s(\beta_j)$ and $T_e(\alpha) < T_e(\beta_j)$ where T_s (resp. T_e) is a function that takes an occurrence of an episode as an input and returns its starting (resp. ending) time. The set of all valid occurrences of an episode rule $\alpha \Rightarrow \beta$ is denoted by $occER(\alpha \Rightarrow \beta)$.

For example, consider the sequence shown in Figure 3 and the support threshold $minsup = 3$. We calculate the set of frequent episodes with respect to *minsup*. We start with episodes of size 1. For $\alpha = a$, the maximal set of distinct occurrences in sequence S is $do_\alpha = \{[1], [3], [4], [7]\}$; hence, the support of α is 4. Next, we determine the maximal set of distinct occurrences of $\beta = b$, $\gamma = c$, and so on. Then, by joining the timestamp of each occurrence of any pair of episodes according to Definition 6, we obtain the occurrences of larger episodes.

Based on the concept of episode rule occurrence, the support of an episode rule $\alpha \Rightarrow \beta$ is defined as the number of all valid occurrences in the sequence.

Definition 11 (Episode Rule Support) The support of an episode rule $\alpha \Rightarrow \beta$, denoted by $supER(\alpha \Rightarrow \beta)$ is defined as follows:

$$supER(\alpha \Rightarrow \beta) = \|occER(\alpha \Rightarrow \beta)\|$$

The confidence of an episode rule is defined as in a previous work, that is, as the ratio between the support of that episode rule and the support of its antecedent.

Table 1 Frequent episodes with $minsup = 3$

Episode	Occurrences	Support
a	{[1], [3], [4], [5], [7]}	4
b	{[1], [4], [7]}	3
c	{[2], [4], [6]}	3
d	{[2], [3], [7]}	3
ab	{[1 1], [3 4], [5 7]}	3
abc	{[1 1 2], [3 4 4], [5 7 6]}	3
$abcd$	{[1 1 2 2], [3 4 4 3], [5 7 6 7]}	3
ac	{[1 2], [3 4], [5 6]}	3
acd	{[1 2 2], [3 3 3], [4 7 7]}	3
ad	{[1 2], [3 3], [4 7]}	3
bc	{[1 2], [4 4], [7 6]}	3
bcd	{[1 2 2], [4 4 3], [7 6 7]}	3
bd	{[1 2], [4 3], [7 7]}	3
cd	{[2 2], [4 3], [6 7]}	3

Definition 12 (Episode Rule Confidence) The confidence of an episode rule $\alpha \Rightarrow \beta$ is denoted by $conf(\alpha \Rightarrow \beta)$ and it is defined as follows:

$$conf(\alpha \Rightarrow \beta) = \frac{\|occER(\alpha \Rightarrow \beta)\|}{support(\alpha)}$$

In addition to the set of frequent episodes already demonstrated, a set of episode rules can be derived in a straightforward manner. Consider the confidence threshold $minconf = 50\%$. For instance, let $\alpha = a$ and $\beta = d$ be two frequent episodes, each identified by its distinct occurrence, as shown in Table 1. According to Definition 10, we can obtain the set of occurrences of $ER = a \Rightarrow d$ as $occER(a \Rightarrow d) = \{[1\ 2], [3\ 7]\}$; hence, the support of the rule is $suppER(a \Rightarrow d) = \|occER(a \Rightarrow d)\| = 2$. Therefore, the confidence is straightforwardly calculated by definition 12 as $conf(a \Rightarrow d) = \frac{\|occER(a \Rightarrow d)\|}{support(a)} = \frac{2}{4} = 0.5 = 50\%$. Table 2 lists a subset of episode rules derived from sequence 3 with respect to a confidence threshold $minconf = 50\%$.

The problem addressed in this paper is the mining of frequent episodes and episode rules using the distinct occurrence-based frequency. More precisely, given a sequence S, a support threshold $minsup$ and a confidence threshold $minconf$, the proposed approach consists of two tasks:

Table 2 Frequent episodes with $minconf = 50\%$

Episode Rule $\alpha \Rightarrow \beta$	Occurrences	conf($\alpha \Rightarrow \beta$)
$a \Rightarrow d$	{[1 2], [3 7]}	50%
$a \Rightarrow dc$	{[1 2 2], [3 6 7]}	50%
$a \Rightarrow dcb$	{[1 3 4 4], [3 7 6 7]}	50%
$a \Rightarrow dcba$	{[1 3 4 4 3], [3 7 6 7 5]}	50%
$a \Rightarrow c$	{[1 2], [3 4], [4 6]}	75%
$a \Rightarrow cb$	{[1 4 4], [3 6 7]}	50%
$a \Rightarrow cba$	{[1 4 4 3], [3 6 7 5]}	50%
$a \Rightarrow b$	{[1 4], [3 7]}	50%
$a \Rightarrow ba$	{[1 4 3], [3 7 5]}	50%
$a \Rightarrow a$	{[1 3], [3 4], [4 5]}	75%
$da \Rightarrow d$	{[2 1 3], [3 3 7]}	66.67%
$da \Rightarrow dc$	{[2 1 3 4], [3 3 7 6]}	66.67%
$da \Rightarrow dcb$	{[2 1 3 4 4], [3 3 7 6 7]}	66.67%
$da \Rightarrow dcba$	{[2 1 3 4 4 3], [3 3 7 6 7 5]}	66.67%

- Finding all frequent episodes under distinct occurrences-based frequency, i.e., having a support which is greater or equal to *minsup*.
- Finding all valid episode rules of the form $\alpha \Rightarrow \beta$ such that α and β are frequent episodes and $conf(\alpha \Rightarrow \beta) \geq minconf$.

4 Novel efficient algorithms for episode rule mining in complex sequences

This section presents the proposed approach for mining frequent episodes and episode rules under distinct occurrences-based frequency. The proposed approach involves two main phases: the first phase consists of mining frequent episodes using a procedure to **recognize distinct occurrences**, whereas the second phase extracts all valid episode rules in the proposed form, as discussed in the previous section.

4.1 Frequent episode mining with distinct occurrences-based frequency

The first step is to mine the set of all frequent episodes according to their distinct occurrences. Initially, the function **Mine_frequent_episodes**, presented

in **Algorithm 1**, scans the input sequence to extract episodes of size 1 (containing a single event). For each event type e that occurs at a timestamp t_k, t_k is added to a variable do_e that stores the distinct occurrences of e. Next, the function retains only frequent event types (line 1-9). Then, the algorithm mines larger episodes using a depth-first search strategy by repeatedly combining each n-node episode with a single event episode to form larger episodes. Function **Distinct_Occurrence_Recognition** (Algorithm 2) is used to find the maximal set of distinct occurrences of each produced episode.

To avoid exploring all of the search space, the function for frequent episode generation under distinct occurrences-based support utilizes the following anti-monotonicity property.

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

Proof Since $\alpha \sqsubseteq \beta$, it follows that each occurrence of β in S includes an occurrence of α. Hence, the number of occurrences of episode β is at most equal to the number of occurrences of episode β, i.e. $support(\alpha) \geq support(\beta)$. Therefore, if the episode β is frequent, i.e., $support(\beta) \geq minsup$ and since $support(\beta) \leq support(\alpha)$ then $support(\alpha) \geq minsup$, hence, episode α is also frequent. Consequently, the anti-monotonicity property holds for the distinct occurrences-based support. □

Function **Distinct_Occurrence_Recognition**, which finds the maximal set of distinct occurrences (**Algorithm 2**), receives two input episodes: an episode α to be grown using a single event episode *beta*. The output is the distinct occurrences of the resulting episode $do_{\alpha \sqcup \beta}$. Several FEM algorithms perform multiple scans of the input sequence to calculate the occurrences of an episode. However, these scans generally consume excessive time and memory. The proposed algorithm avoids this problem by calculating the occurrences of any new larger episode of size $n+1$ starting from the set of occurrences of episodes of size n and that of a single event episode.

For each occurrence, $O_i \in do_\alpha$, the algorithm parses the set of distinct occurrences of β to obtain an occurrence $O_j \in do_\beta$ and builds an occurrence of $\alpha \sqcup \beta$ whose vector of timestamps contains timestamps of O_i (i.e., $O_i.timestamps$ from do_α) joined with timestamps of O_j (i.e., $O_j.timestamps$ from do_β).

Then, the algorithm compares the new occurrence with all the occurrences in the set $do_{\alpha \sqcup \beta}$; if that new occurrence overlaps with any other occurrence in the current maximal set of distinct occurrences of $\alpha \sqcup \beta$ (i.e., the set $do_{\alpha \sqcup \beta}$), the algorithm simply checks which occurrence it is to remove from the set do_α or from do_β; if the timestamp of the event in β (here, since $\|\beta\| = 1$, $O_j.timestamps[1]$ is the occurrence time of the single event episode β in S) exists in the vector $newocc.timestamps$, then O_j is removed from do_β. Otherwise, one timestamp from $O_i.timestamps$ surely exists in

*newocc.timestamps*j; in this case, O_i is removed from do_α (line 10-16). Here, the function $exists(t_k, O_i)$ returns *true* if the integer (timestamp) t_k exists in the vector of times $O_i.timestamps$; otherwise, it returns $false$.

4.1.1 An illustrative example

Consider the complex sequence of Fig 3 as an example, and let $minsup = 3$. In the first step of Algorithm 1, each event type in E is viewed as a single event episode (i.e., an episode with one event). Next, the algorithm computes, for each episode, the set of its occurrences in each event set by scanning the complex event sequence, and the support according to definition 8 (see line 1-14). Then, the algorithm removes the non-frequent single event episodes based on the support threshold (see line 15-19). Table 3 shows the set P of frequent episodes obtained by executing lines 1-19.

Table 3 Frequent episodes of size 1 with $minsup = 3$

Episode	Occurrences	Support
a	$\{[1], [3], [4], [7]\}$	4
b	$\{[1], [4], [7]\}$	3
c	$\{[2], [4], [6]\}$	3
d	$\{[2], [3], [7]\}$	3

To discover larger episodes, the algorithm creates a copy of the frequent episodes already obtained and then starts the search for larger episodes, following a depth-first strategy. Before deciding whether a new episode $\alpha \sqcup \beta$ is frequent, the algorithm computes the episode's distinct occurrences denoted by $do_{\alpha \sqcup \beta}$. An example of this process will be given after. Next, the algorithm checks if the episode's support meets the requirement. If so, the episode is added into the set F of frequent episodes and the algorithm continues the search (line 21-28). To perform the step of *distinct occurrences recognition*, the algorithm 1 calls Algorithm 2. Consider two episodes $\alpha = a$ and $\beta = b$ where $do_\alpha = \{[1], [3], [4], [7]\}$ and $do_\beta = \{[1], [4], [7]\}$. The first step of each iteration in Algorithm 2 within the loop is to create a new occurrence and initialize its timestamps as the union between the timestamps of an occurrence of α and those of the occurrence of β. As explained before, the first occurrences $O_\alpha = [1]$ and $O_\beta = [1]$ will result in an occurrence of $\alpha \sqcup \beta$ such that $newocc_{\alpha \sqcup \beta}.timestamps = [1\ 3]$. The algorithm continues for the next occurrences $O_\alpha = [3]$ *and* $O_\beta = [4]$ such that $newocc_{\alpha \sqcup \beta}.timestamps = [3] \cup [4] = [3\ 4]$ and stores each occurrence in $do_{\alpha \sqcup \beta}$. However, there is an exceptional case where an occurrence of $\alpha \sqcup \beta$ does not meet the criteria of definition 6 (line 11). In this case, the process uses another way of selectingd which occurrence to overstep. Therefore, if the timestamp of the single

event episode β exists already in any old occurrence of $\alpha \sqcup \beta$ then, the algorithm loops with the same occurrence of α and selects the next occurrence of β. Otherwise, the algorithm keeps the occurrence of β and tests the combination with the next occurrence of α since it absolutely intersects with any other occurrence of $\alpha \sqcup \beta$. For instance, consider two episodes $\alpha = a$ and $\beta = b$. If the algorithm has the occurrence $O_\alpha = [3]$ and $O_\beta = [4]$ then, $do_{\alpha \sqcup \beta} = [3\ 4]$. However, the next combination between $O_\alpha = [4]$ and $O_\beta = [7]$ will not be valid since the timestamp $t = 4$ already exists in $[3\ 4]$. Hence, the algorithm simply keeps the occurrence $O_\beta = [7]$ and moves to the next occurrence $O_\alpha = [5]$, which gives a valid occurrence of $\alpha \sqcup \beta$ such that: $do_{\alpha \sqcup \beta} = \{[1\ 1], [3\ 4], [5\ 7]\}$. When the algorithm terminates, all frequent episodes have been found. Table 1 shows the final set of frequent episodes generated by Algorithm 1 for the sequence of Fig 3.

4.2 Episode Rule Mining

The following paragraphs present an extension of the algorithm proposed in Section 4.1 to derive all valid episode rules of the form $\alpha \Rightarrow \beta$ where α and β are two frequent episodes under the distinct occurrences-based frequency.

The episode rule mining process is given by the function **Extract_Episode_Rules** described in Algorithm 3 which takes as input a complex event sequence S, support threshold $minsup$ and confidence threshold $minconf$ where the output is the set R of the valid episode rules.

Initially, the function calculates the set of frequent episodes by calling the **Mine_Frequent_Episodes** function (Algorithm 1), and initializes the set of rules (lines 1-2). Then, it iterates on the set of frequent episodes to capture valid rules using Definition 11. Here, the function **Episode_Rule_Support(α, β)** (line 5) calculates the support of the rule according to **Definition 11** (See Algorithm 4)

4.3 Episode Rule mining with pruning

The approach presented in the previous subsection allows finding all episode rules. However, considering all possible combinations of α and β to form an episode rule leads to a large search space. To reduce this search space, a pruning technique is applied as follows: For a given single event episode used as a consequent of an episode rule, only its super-episodes are candidates to be consequents of valid episode rules. This is obtained using the anti-monotonicity property in the rule generation step, as stated by Proposition 2 below.

Proposition 2 *Let α and β be two frequent episodes under distinct occurrences-based frequency. If the rule $\alpha \Rightarrow \beta$ is invalid, then every episode rule $\alpha \Rightarrow \gamma$ such that $\beta \sqsubseteq \gamma$ is invalid too.*

Algorithm 1 Mine_Frequent_episodes

Require: *minsup* - a minimum support threshold
Ensure: F - the set of all frequent parallel episodes

$P \Leftarrow \{\}$;
$F \Leftarrow \{\}$;
$\alpha \Leftarrow \emptyset$;
for (each individual event $e \in E$) **do**
    $do_e \Leftarrow \{\}$
end for
{% scan the sequence S }
for ($k \Leftarrow 1$ to n) **do**
    {% scan the event set found at time t_k %}
    for ($j \Leftarrow 1$ to m) **do**
        $e_j \Leftarrow$ the event found at time t_k;
        $do_{e_j} \Leftarrow do_{e_j} \cup \{t_k\}$;
    end for
end for
for (each individual event $e \in E$) **do**
    if ($\|do_e\| \geq minsup$) **then**
        $P \Leftarrow P \cup \{(e)\}$;
    end if
end for
$F \Leftarrow P$;
for (each individual episode $\alpha \in F$) **do**
    for (each individual episode $\beta \in P$) **do**
        $do_{\alpha \sqcup \beta} \Leftarrow$ **Distinct_Occurrence_Recognition**(α, β);
        if ($\|do_{\alpha \sqcup \beta}\| \geq minsup$) **then**
            $\gamma \Leftarrow \alpha \sqcup \beta$;
            $do_\gamma \Leftarrow do_{\alpha \sqcup \beta}$;
            $F \Leftarrow F \cup \{\gamma\}$;
            $\alpha \Leftarrow \gamma$;
        end if
    end for
end forreturn F;

Proof Proving the correctness of that proposition simply requires to use the anti-monotonicity of the distinct occurrences-based support, to show that the confidence of a rule $\alpha \Rightarrow \gamma$ obtained from a single event episode β such that $\beta \sqsubseteq \gamma$, will be invalid. Let $\beta \sqsubseteq \gamma$ be two episodes, and assume that the support of the rule $\alpha \Rightarrow \beta$ is invalid. By the anti-monotonicity property in Proposition 1, $support(\beta) \geq support(\gamma)$. It follows that :

$$conf(\alpha \Rightarrow \beta) = \frac{\|occER(\alpha, \beta)\|}{support(\alpha)} \geq conf(\alpha \Rightarrow \gamma) = \frac{\|occER(\alpha, \gamma)\|}{support(\alpha)}$$

. However, since the rule $\alpha \Rightarrow \beta$ is invalid, this means that $conf(\alpha \Rightarrow \beta) < minconf$, and therefore, the rule $\alpha \Rightarrow \gamma$ is invalid as well. □

Algorithm 2 Distinct_Occurrence_Recognition

Require: episode α - an episode to grow episode β - a single event episode to be used to grow α.
Ensure: $do_{\alpha \sqcup \beta}$ - the set of distinct occurrences of the new episode $\alpha \sqcup \beta$

1: $j \Leftarrow 0$
2: $i \Leftarrow 0$
3: **for** (each $O_i \in do_\alpha$) **do**
4: $found \Leftarrow false$
5: **for** (each $O_j \in do_\beta$ and $found = false$) **do**
6: $newocc.timestamps \Leftarrow O_i.timestamps \cup O_j.timestamps$
7: $stop \Leftarrow$ **false**
8: $k \Leftarrow 1$
9: **while** (**not** $stop$ **and** $k \leq \|do_{\alpha \sqcup \beta}\|$) **do**
10: **if** ($newocc.timestamps \cap O_k.timestamps \neq \emptyset$) **then**
11: **if** ($exists(O_j.timestamps[1], O_k.timestamps) =$ **true**) **then**
12: remove O_j from do_β
13: **else**
14: remove O_i from do_α;
15: **end if**
16: $stop \Leftarrow true$
17: **end if**
18: $O_k \Leftarrow O_{k+1}$
19: **end while**
20: **if** (**not** $stop$) **then**
21: $do_{\alpha \sqcup \beta} \Leftarrow do_{\alpha \sqcup \beta} \cup newocc$
22: $found \Leftarrow$ **true**
23: **end if**
24: **end for**
25: **end for**
26: **return** $do_{\alpha \sqcup \beta}$

Algorithm **Extract_Episode_Rules_With_Pruning** (Algorithm 5) integrates a pruning strategy based on Proposition 2 to generate all valid episode rules from the input complex-event sequence.

Initially, the algorithm calculates the set of all frequent episodes with respect to the support threshold $minsup$, initializes the set R of valid rules to be calculated, and initializes the set P of frequent single episodes (lines 1-3). Then, for each frequent episode α as an antecedent of rule $\alpha \Rightarrow \beta$, the algorithm calculates the set of valid consequents with respect to the confidence threshold $minconf$ by combining larger and larger episodes with single event episodes. First, the algorithm uses the j^{th} single event as a consequence of the episode rule. Here, the episode $root$ in line 8 is used to backtrack the process if there is no other candidate super-episode of β as a consequence. For each episode β, the algorithm calculates the episode rule support by calling

Algorithm 3 Extract_Episode_rules

Require: S: complex event sequence on E (the set of event types) ,$minsup$: support threshold ,$minconf$: confidence threshold
Ensure: R: complete set of episode rules

1: $F \Leftarrow$ **Mine_Frequent_Episodes**($minsup$)
2: $R \Leftarrow \emptyset$
3: **for** (each α in F) **do**
4: **for** (each β in F) **do**
5: $ruleSupport \Leftarrow$ **Episode_Rule_Support**(α, β);
6: **if** $(\frac{ruleSupport}{\|do_\alpha\|} \geq minconf)$ **then**
7: $R \Leftarrow R \cup \{\alpha \Rightarrow \beta\}$
8: **end if**
9: **end for**
10: **end for**
11: **return** R

the function $Episode\_Rule\_Support(\alpha, \beta)$ and calculates its confidence: If the confidence exceeds the confidence threshold, the algorithm adds the episode $\alpha \Rightarrow \beta$ into R and updates the *root* episode for the later combination of the consequents (lines 11-19). The algorithm stops if there are no other possible single-event episode candidates; otherwise, it updates episode β to become larger with the next j^{th} episode from P (lines 20-23).

4.3.1 An illustrative example

An example is presented to illustrate the process of discovering the set of all episode rules in the sequence S given in Fig. 3 for $minsup = 3$ and $minconf = 50\%$ by applying Algorithm 5. Initially, the algorithm calculates the set of frequent episodes as explained before (see Table 1). Then, it creates the set P of frequent episodes of size 1 (line 3) from the sequence S, i.e., $P = \{a, b, c, d\}$. Next, the algorithm will search for episode rules. Take episode $\alpha = a$, the algorithm finds all rules where the antecedent is $\alpha = a$ ($\alpha \in F$) and then, it considers $\beta = b$ ($\beta \in P$) as the first consequent and the root of potential consequents (line 4 -10), where $do_\alpha = \{[1], [3], [4], [5], [7]\}$ $do_\beta = \{[1], [4], [7]\}$. Then, the algorithm calculates the support of the episode rule $ER = \alpha \Rightarrow \beta$ (line 13) according to definition 10 and 11. Here, the support is calculated as follows: for each occurrence $O_i \in do_\alpha$, Algorithm 4 is applied to find an occurrence $O_j \in_\beta$ where $do_\alpha = \{[1], [3], [4], [7]\}$ and $do_\beta = \{[1], [4], [5]\}$ such that $start(O_i) < start(O_j)$ and $end(O_i) < end(O_j)$. Hence, for the episode rule $ER = \alpha \Rightarrow \beta$, its first occurrence is [1 4] and cannot be [1 1] since the occurrence $O_\alpha = [1]$ and $O_\beta = [1]$ do not meet the previous criteria. Thus, the algorithm jumps to the second occurrence of β ($O_\beta = [4]$), which produces a valid occurrence of the episode rule. Then, the algorithm moves to the next occurrences $O_\alpha = [3]$ and $O_\beta = [5]$ of α and β respectively, which make the vector [3 5] a valid occurrence of the episode rule. The set of occurrences of

Algorithm 4 Episode_Rule_Support

Require: Two Episodes α and β
Ensure: *ruleSupport*: The support of the rule $\alpha \Rightarrow \beta$

```
1: {% Inialization %}
2: ruleSupport ⇐ 0;
3: i ⇐ 1;
4: j ⇐ 1;
5: while (i ≤ ||do_α||) do
6:     stop ⇐ false;
7:     while (j ≤ ||do_β|| and not stop) do
8:         for (each O_k in do_β s.t: j < k ≤ ||do_β||) do
9:             if (start(O_i) < start(O_j) and end(O_i) < end(O_j)) then
10:                i ⇐ i + 1;
11:                ruleSupport ⇐ ruleSupport + 1;
12:                stop ⇐ true;
13:            end if
14:        end for
15:        j ⇐ k;
16:    end while
17: end while
18: return ruleSupport ;
```

the rule $\alpha \rightarrow \beta$ is $ER - occ(\alpha \Rightarrow \beta) = \{[1\ 4], [3\ 7]\}$ Finally, the support is calculated as $supER(\alpha \Rightarrow \beta) = \|occER(\alpha \Rightarrow \beta)\| = 2$. Then, Algorithm 5 continues its process by calculating the confidence on line 14. If the confidence exceeds the $minconf$ threshold, then, the current rule is considered as valid and the current consequent β becomes the root of potential consequents such that it is joined with the next single event episode $\gamma = c$ from the set P and the process is repeated. Notice that, if the consequent is not frequent, the root takes the place of the valid consequent to join. For $minconf = 50\%$, the rule $a \Rightarrow \beta$ is valid and $\beta = b$ is the root for future consequents, hence, it is joined with episode $\gamma = c$. Therefore, the next episode rule to check is $a \rightarrow bc$. if the last rule is not valid with respect to $minconf$, $\beta = b$ will be joined with $\delta = d$ and so on according the proposition 2. Table 2 shows the final set of valid episode rules found by the algorithm.

5 Experimental study

Several experiments were conducted on both synthetic and real datasets to evaluate the proposed approach for mining frequent episodes and episode rules. The experiments were performed on an AMD Ryzen 5 PRO 4650G with Radeon Graphics 3.70 GHz PC with 16 Gb of main memory and 256 Gb of SSD storage, running the Microsoft Windows 10 operating system. All algorithms were coded in Java. These experiments only evaluates the designed approach

Algorithm 5 Extract_Episode_Rules_With_Pruning

Require: S: complex event sequence on E (the set of event types) $minsup$: support threshold $minconf$: confidence threshold
Ensure: R: complete set of episode rules

```
1: F ⇐ Mine_Frequent_Episodes(minsup)
2: R ⇐ ∅
3: P ⇐ {γ s.t. γ ∈ F ∧ ‖γ‖ = 1}
4: for (each α ∈ F) do
5:     j ⇐ 0
6:     while (j < ‖P‖) do
7:         β ⇐ P[j]
8:         root ⇐ β
9:         k ⇐ j
10:        stop ⇐ false
11:        while (not stop) do
12:            if (β ∈ F) then
13:                ruleSupport ⇐ Episode_Rule_Support(α, β)
14:                conf ⇐ ruleSupport / ‖alpha.occ‖
15:                if (conf ≥ minconf) then
16:                    R ⇐ R ∪ {α → β}
17:                    root ⇐ β
18:                else
19:                    β ⇐ root
20:                end if
21:            else
22:                β ⇐ root
23:            end if
24:            if (k > ‖P‖) then
25:                stop ⇐ true
26:            else
27:                β = β ⊔ P[k]
28:                k ⇐ k + 1
29:            end if
30:        end while
31:        j ⇐ j + 1
32:    end while
33: end for
34: return R
```

since there exist no prior work on discovering frequent episodes in complex event sequences using the distinct occurrences-based frequency definition.

5.1 Data generation

Several synthetic datasets were used for the experiments, which were randomly generated using three main parameters:(i) the length of the complex sequence (the number of events), (2) the number of event types, and (3) the maximal size of event sets in the complex sequence. To evaluate the proposed approach under different scenarios, three types of synthetic sequences were considered: short, medium, and large. Table 4 lists the details of the chosen synthetic datasets for these experiments. The datasets are available publicly on GitHub [1]

Table 4 Synthetic datasets parameters

Type	Number of event types	sequence size (number of event sets)
Small dataset	15	40000
Medium dataset	15	65000
Large dataset	20	70000

Three real datasets were obtained in the form of transaction databases from the SPMF dataset collection [2]. For this purpose, each item is considered an event, and hence each transaction (itemset) is considered as an event set in the complex sequence. Furthermore, each event set is associated with an integer (the transaction number) to represent its timestamp. The first dataset that was used is called *OnlineRetail* with 541880 event sets and 2603 event types, the second dataset is called *FruitHut* with 181970 event sets and 9390 event types, and the last one is called *Mushrooms* which contains 8416 event sets and 119 event types. These three real datasets were chosen owing to their different characteristics.

5.2 Results and discussion

The experiments described in this section aim to evaluate the performance of two processes: mining frequent episodes (Section 5.2.1) and generating valid episode rules (Section 5.2.2).

For frequent episode mining, the performance analysis considers both synthetic and real datasets and the variation according to the support threshold of: (1) the runtime (in seconds), (2) the number of frequent episodes, (3) the size of the largest frequent episode, and (4) the memory used during the process.

For episode rule mining, the baseline algorithm (called EMDO) was compared with the algorithm for mining episode rules using the pruning strategy (called EMDO-P), both proposed in this paper, in terms of (1) runtime and (2) memory cost for both synthetic and real datasets.

[1] https://github.com/Oualidinx/EMDO-synthetic-datasets.git
[2] http://www.philippe-fournier-viger.com/spmf/, last accessed on 2022-12-01

5.2.1 Frequent episode mining

The first step of the proposed algorithm is to generate a set of frequent episodes using Algorithm 1. Figure 5 and 6 show how the support threshold influences the number of frequent episodes and maximum episode size on synthetic and real datasets, respectively.

The variations in the number of frequent episodes with respect to the support threshold values clearly show that the anti-monotony property of the distinct occurrence-based frequency is very effective for pruning the search space by the proposed approach on both synthetic and real datasets. Moreover, the size of the largest frequent episode is much smaller when the minimum support value is increased. The smaller *minsup* values are, the more frequently large episodes occur, and the greater the number of frequent episodes is for synthetic or real datasets.

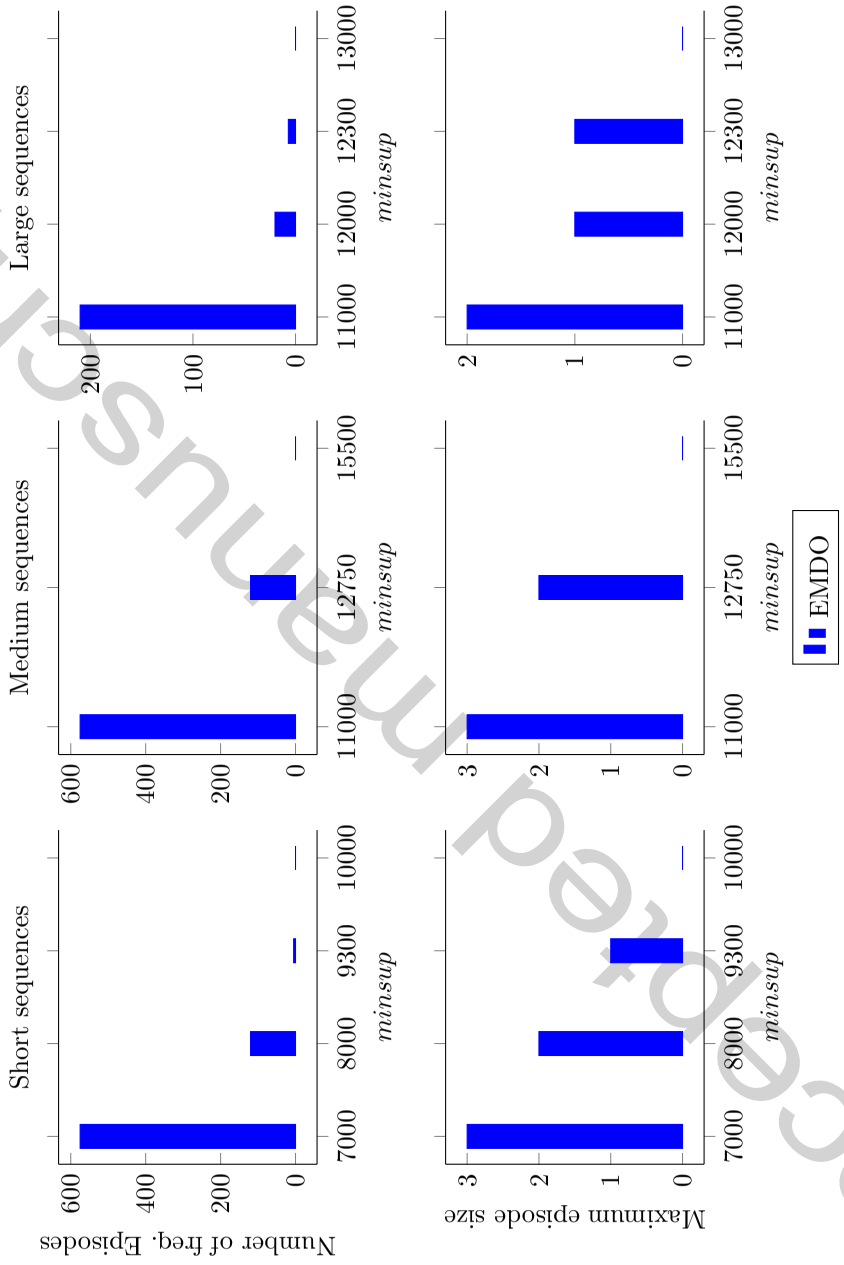


Fig. 5 Influence of *minsup* on the number of frequent episodes and the size of the largest episode on synthetic datasets

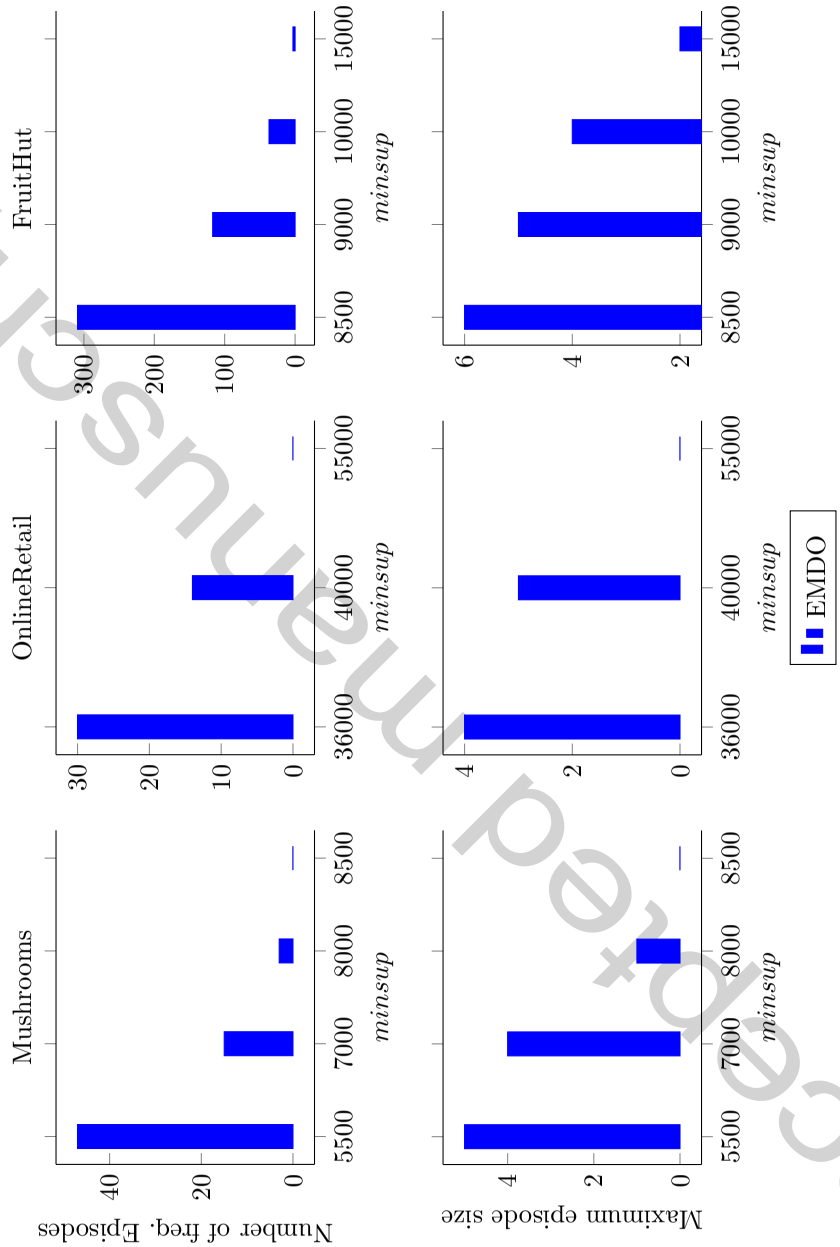


Fig. 6 Influence of *minsup* on the number of frequent episodes and the size of the largest episode on real datasets

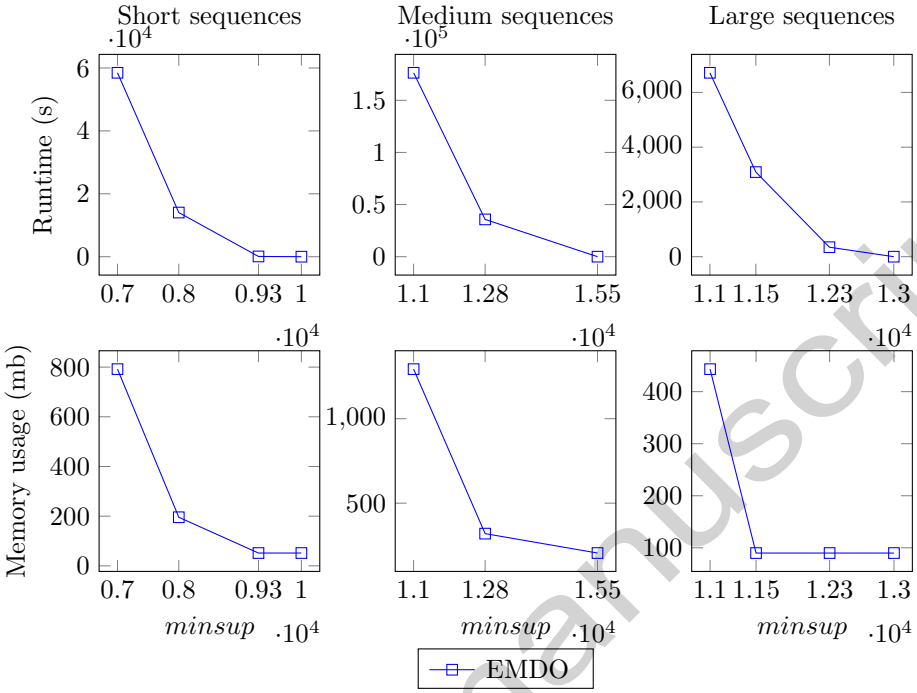


Fig. 7 Influence of *minsup* for frequent episode generation in terms of execution time and memory usage on synthetic data sets

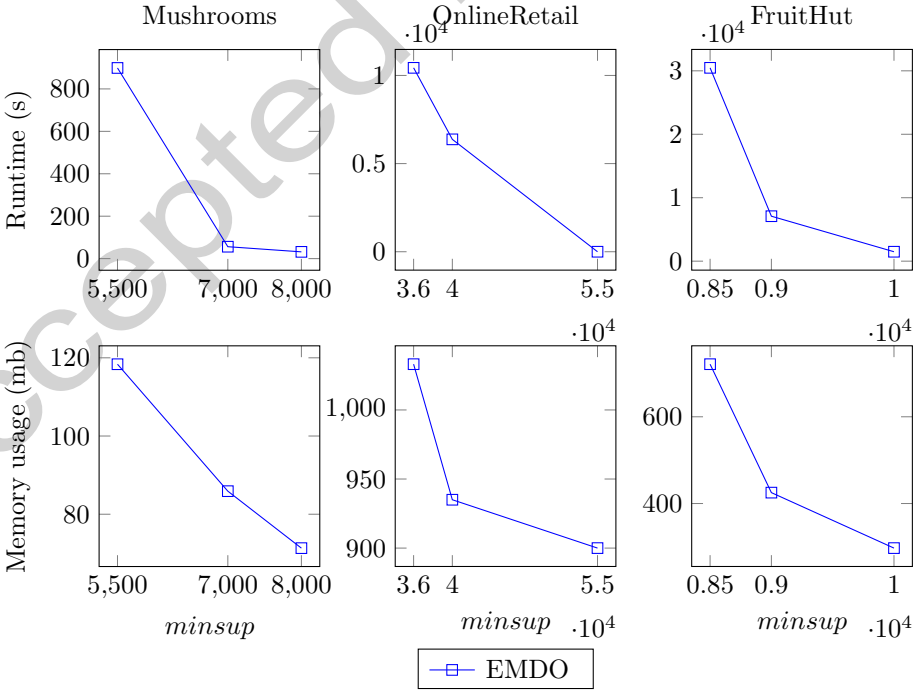


Fig. 8 Influence of *minsup* on execution time and memory usage on real data sets

Figures 7 and 8 depict the results obtained from the application of the proposed EMDO algorithm for different *minsup* threshold values in terms of runtime (in seconds) and memory usage (in megabytes) for synthetic sequences (short, medium and large size) and real sequences, respectively.

The number of frequent episodes with respect to the *minsup* threshold in figures 7 and 8 decrease when the support increase. Moreover, the memory cost also decreases rapidly for greater support thresholds. This shows in particular the efficiency of applying the anti-monotonicity property in the context of distinct occurrences-based frequency. The property states that the support of an episode is not greater than that of any of its subepisodes. Consequently, the greater support threshold values, the less runtime and memory costs. Therefore, the results show the efficiency of the proposed approach in terms of runtime, memory usage, frequent episode count, and episode sizes.

5.2.2 Episode rule generation

The second step of EMDO is the generation of valid episode rules from a derived set of frequent episodes. Because there are no other algorithms that rely on distinct occurrences-based support to mine frequent parallel episodes and/or episode rules in complex event sequences, we focus here on the comparison between the baseline version of the proposed EMDO algorithm and the modified version, (EMDO_P) which utilizes the pruning technique described in Section 4.3. The main objective of this experiment was to evaluate the efficiency of the proposed pruning technique for generating valid episode rules. Note that the two versions (EMDO and EMDO_P) yield the same output, that is, generate the same valid episode rules. However, they differ in performance, as explained below. The different minimum support values are presented in Table 5.

Table 5 The supports value used in the episode rules generation step

Type of dataset	Datasets	Support threshold value
Synthetic datasets	Short dataset	1000
	Medium dataset	1750
	Large dataset	3000
Real datasets	Mushrooms	1050
	FruitHut	5000
	OnlineRetail	10000

Figures 9 and 10 illustrate the influence of the $minconf$ threshold on runtime (in seconds) and memory usage (in megabytes) for the naive algorithm EMDO and the improved algorithm EMDO_P on synthetic and real datasets, respectively.

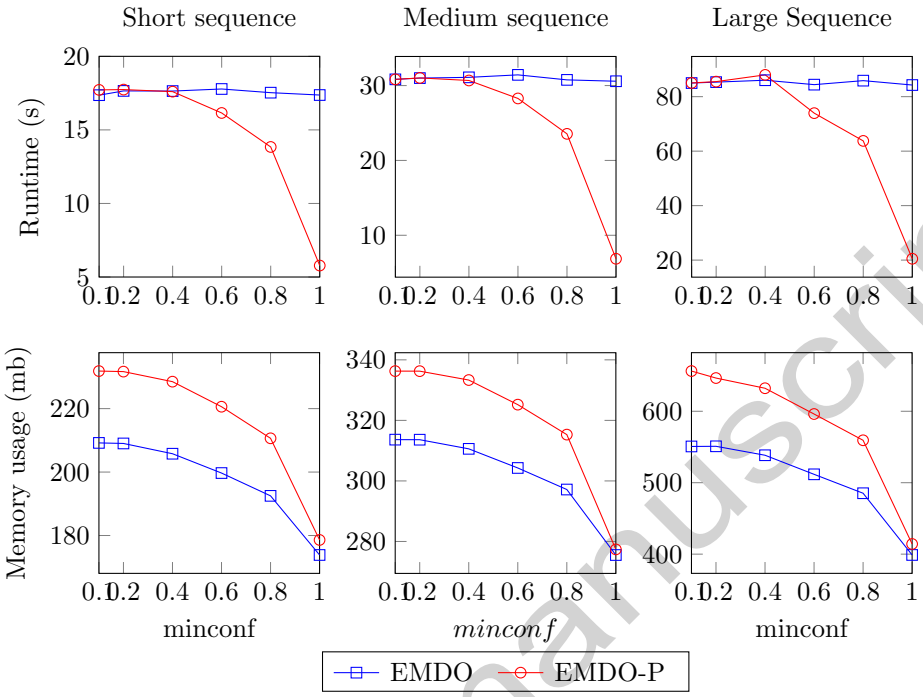


Fig. 9 Influence of *minconf* on execution time and memory usage of episode rules generation on synthetic datasets

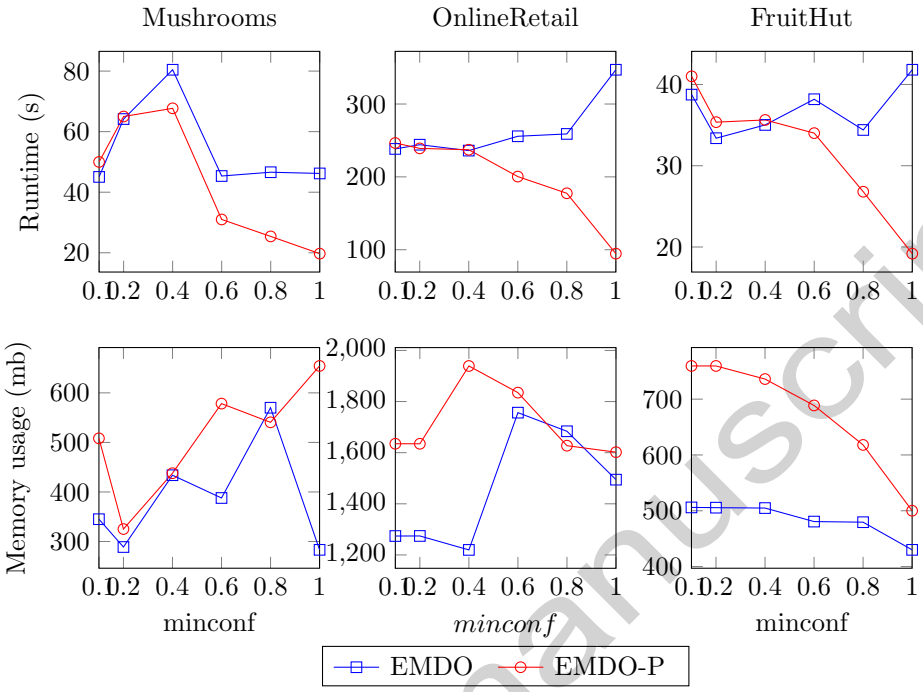


Fig. 10 Influence of *minconf* on execution time and memory usage of episode rules generation on real datasets

Unsurprisingly, the naive algorithm globally uses less memory to generate valid episode rules for different values of the confidence threshold *minconf* than the algorithm using the pruning strategy. This remark is valid both for synthetic and real datasets. This is not problematic because the increase in memory cost remains reasonable and does not challenge the modified algorithm.

However, the modified algorithm EMDO_P is clearly better than the naive EMDO algorithm in terms of runtime for both synthetic and real datasets, particularly when the confidence threshold is increased. This is because the pruning strategy enables EMDO_P to eliminate many large episodes that cannot be a consequent of valid rules. This is due to the new episode rule pruning strategy of our algorithm that avoid the combination of many frequent episodes, when compared to the naive algorithm.

5.3 Discussion of some discovered patterns

Using the EMDO_P algorithm, several patterns can be retrieved from datasets that reveal hidden relationships between events. For instance, we executed the proposed method on the *FruitHut* dataset, and the resulting set of episode rules was significant. As example, Table 6 presents some of the rules extracted from this dataset.

These rules represent strong relationships between purchases made by customers. It is interesting to note that those rules cannot be generated by the NONEPI algorithm because this latter only generate rules where the antecedent is a subepisode (predecessor) of the consequent. However, our approach can combine all episodes to make rules, rather than only those where an antecedent is a sub-episode of a consequent. Moreover, the rules generated by the NONEPI algorithm can be generated using the EMDO_P algorithm. For instance, the rule *Field Tomatoes* $\rightarrow$ *Field Tomatoes*, *Banana Cavendish* was generated by both NONEPI and EMDO_P for a confidence threshold of $minconf = 50\%$. However, NONEPI cannot generate any of the rules shown in Table 6 except for the rule previously mentioned.

Table 6 Example of discovered patterns from *FruitHut* dataset

Rule	$conf(\alpha \rightarrow \beta)$
Banana Cavendish,Cucumber Lebanese $\rightarrow$ Lettuce Iceberg	85.63%
Banana Cavendish,Cucumber Lebanese $\rightarrow$ Lettuce Iceberg, Beans green	68.63%
Lettuce Iceberg $\rightarrow$ Apples Pink Lady, Zucchini green	86.14%
Beans green, Onion Spring $\rightarrow$ Water melon seedless	96.44%
Beans green, Onion Spring, Capsicum red $\rightarrow$ Field Tomatoes	99.98%
Pear Packham $\rightarrow$ Apples Pink Lady, Zucchini green, Mandarin Imperial	83.44%
Field Tomatoes $\rightarrow$ Field Tomatoes, Banana Cavendish	57.08%

These episode rules are interesting as they show the temporal relationships between items such that if some items are bought in some order, then other items on the left side of such a rule will also be bought by such a customer, which reveals the customer's preference or needs. Consequently, these rules can be used to develop marketing strategies based on promotions or recommendations.

Table 7 Example of discovered episodes by different algorithms from *FruitHut* dataset

Episode	MINEPI+	EMDO	NONEPI	MINEPI
Lettuce Iceberg, Banana Cavendish	23736	8972	7138	7582
Field Tomatoes, Banana Cavendish	35890	20220	13323	14862
Cucumber Lebanese, Banana Cavendish	27034	10489	8261	8886
Field Tomatoes, Cucumber Lebanese, Banana Cavendish	16316	10483	/	/

Table 8 Statistics of different algorithms on the *FruitHut* dataset

	MINEPI+	EMDO	NONEPI	MINEPI
Number of candidate episodes	26642	9959	1942	3993
Number of frequent episodes	992	3243	35	1244
Largest frequent episode size	8	6	2	13

On the real *FruitHut* dataset, we further performed a comparison of patterns found by EMDO and other episode mining algorithms, namely the MINEPI+, NONEPI and MINEPI algorithms. MINEPI+ uses a frequency definition called the head frequency whereas NONEPI uses the non-overlapped occurrence-based frequency to calculate the support of episodes and MINEPI uses the minimal occurrence-based frequency.

Table 7 shows the comparison of support values of frequent episodes discovered by all algorithms. The table indicates that the support calculated by the proposed algorithm is greater than that calculated by NONEPI and MINEPI and less than that calculated by MINEPI+. Consequently, we conclude that EMDO can discover the same frequent episodes but by exploring a smaller search space than MINEPI+ and may provide a more reasonable view as it captures more occurrences than NONEPI and MINEPI.

First, the head frequency is greater than any other frequency definitions owing to the duplicate count of the same events with their timestamps for many windows of length k. Therefore, for a given episode, the head frequency increase with the window length. Second, the minimal occurrence of an episode α is a time interval that contains the occurrence of a given episode such that no proper sub-window contains an occurrence of α, which means that any pair of occurrences can be distinct, but they must be minimal; hence, this additional constraint will eliminate such occurrences that are not minimal. Consequently, distinct occurrence-based frequency will be absolutely greater than minimal occurrences-based frequency since there is no constraint of the presence of occurrences in such time intervals except that they do not share common events (timestamps). Finally, the non-overlapping occurrence-based frequency is the smallest frequency for episodes because of the strong elimination due to the condition that occurrences must not overlap, which eliminates the majority of occurrences of any episode.

Furthermore, Table 8 shows the comparison of the number of frequent episodes, the number of candidate episodes and the size of the largest frequent episodes for each algorithm. It is easy to see that the ratio between the number of frequent episodes to the number of candidate episodes is larger for EMDO which shows the efficiency of our new approach relative to other algorithms. In other words, EMDO has to explore less candidates to find each valid pattern on average.

On overall, the experiments show that EMDO can reveal significant episodes in real data. Due to its frequency function, EMDO may provide a

more accurate view compared to using other algorithms, especially those that use occurrence definitions that shares common events.

6 Conclusion

Several algorithms have been developed to identify episodes in an event sequence. Generally, a frequent episode mining algorithm is designed to utilize a frequency function based on a specific definition of episode occurrence. Although, several algorithms for episode mining have been proposed, most of them consider serial episodes or simple event sequences with only one event per timestamp. Furthermore, most algorithms allow for occurrences to overlap, which can result in counting the same events multiple times, whereas algorithms based on non-overlapping occurrences tend to underestimate the frequency of the episodes.

Based on these observations, this paper studied the problem of episode rule mining in a complex event sequence using distinct occurrences-based frequency. An efficient depth-first strategy was proposed for discovering frequent parallel episodes and valid episode rules, which were integrated into two efficient algorithms, named EMDO and EMDO_P, respectively. To the best of our knowledge, this is the first work that identifies parallel episodes with a distinct occurrences-based frequency in a complex event sequence.

In addition to mining frequent episodes, episode rules represent another important type of pattern to mine from the temporal sequences. An episode rule reveals a strong relationship between frequent episodes that may be used for different practical purposes, such as prediction and diagnosis. In this paper, we propose an adapted strategy for pattern recognition and use it to propose two approaches for extracting episode rules from complex sequences. The first is a naive approach that explores the totality of the search space. The second approach exploits the anti-monotonicity property of the support under the distinct occurrences-based frequency to propose a new pruning strategy for reducing the explored part of the search space. Our pruning strategy is based on the fact that if an episode rule $\alpha \rightarrow \beta$ is not valid with respect to a confidence threshold, then any episode rule $\alpha \rightarrow \beta'$ with $\beta \sqsubseteq \beta'$ is invalid. Consequently, as soon as the algorithm recognizes a non-valid episode rule $\alpha \rightarrow \beta$, it stops the search process for any rule with the form $\alpha \rightarrow \beta'$ such that β' is a super-episode of β.

To demonstrate the performance of our techniques, we performed multiple tests on synthetic and real-world datasets. For synthetic datasets, we built a generator that produces random complex event sequences based on three keys: (i) the length of the result sequence, (ii) the number of event types, and (iii) the maximum number of events per timestamp. For real-world sequences, we used existing real-world transactional databases and added a timestamp to each transaction to obtain a complex sequence. The obtained results confirm the efficiency of the proposed algorithm in terms of both the runtime and memory usage. In particular, they demonstrated the efficiency of the proposed pruning

strategy based on the anti-monotonicity property in discovering episode rules under distinct occurrence-based frequency.

Episode mining and related pattern mining have been active research fields in recent decades. There are still many opportunities for further research in this area. In the following lines, we provide some opportunities for episode and episode rule mining:

- Building prediction models that are based on existing techniques including our approach to be used in production environments .
- Extend our approach to consider uncertain data and/or other kind of important episodes like high utility episodes. This undoubtedly leads to the exploration of new techniques and strategies for pruning search spaces.
- Extending our approach to consider more complex event sequences like event streams .

Data availability

The real and synthetic datasets used in this paper can be downloaded from: https://github.com/Oualidinx/EMDO-synthetic-datasets.git and http://www.philippe-fournier-viger.com/spmf/, respectively.

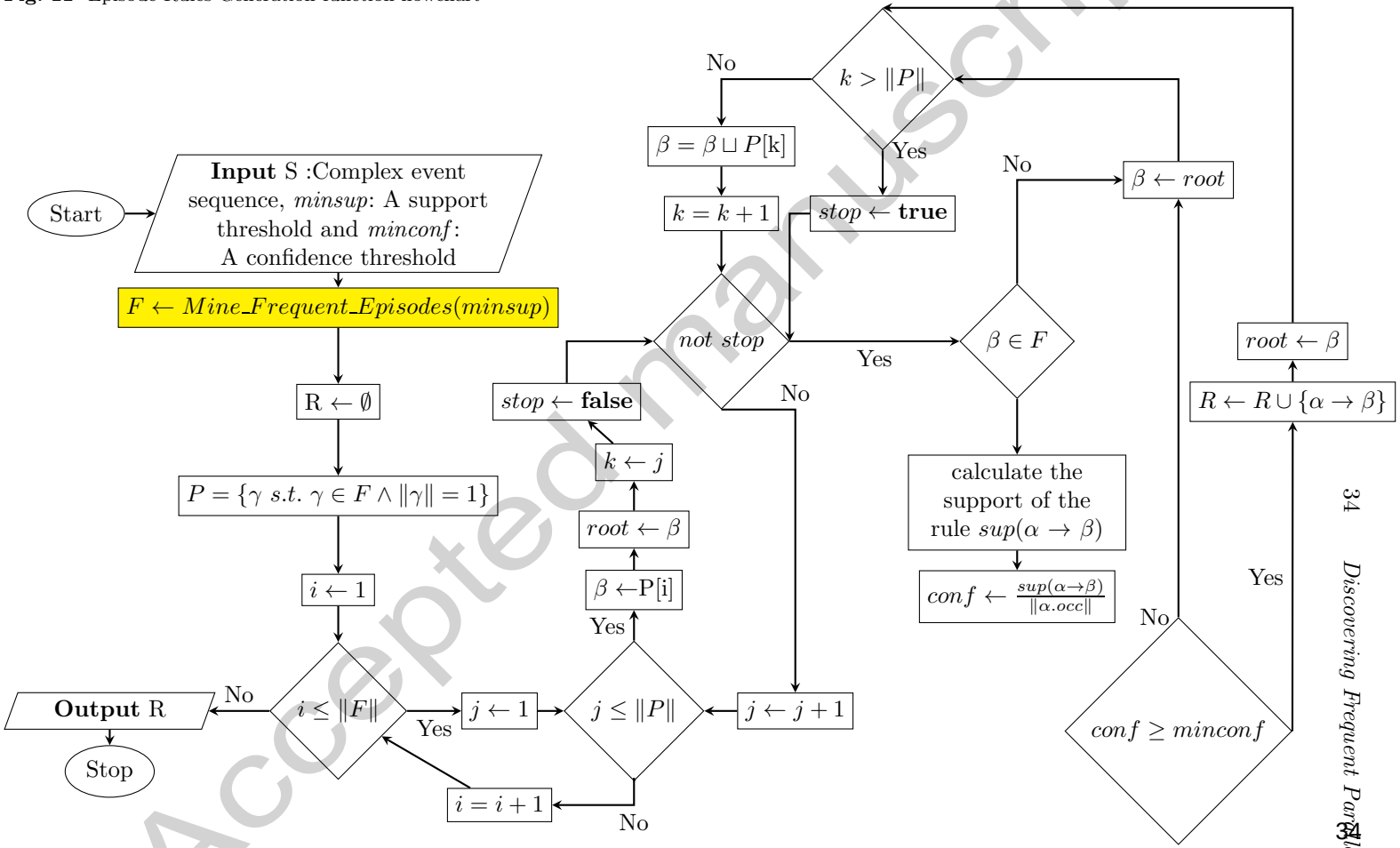
Funding

No funding was received for conducting this study.

Competing interests

The authors have no relevant financial or non-financial interests to disclose.

Fig. 11 Episode Rules Generation function flowchart



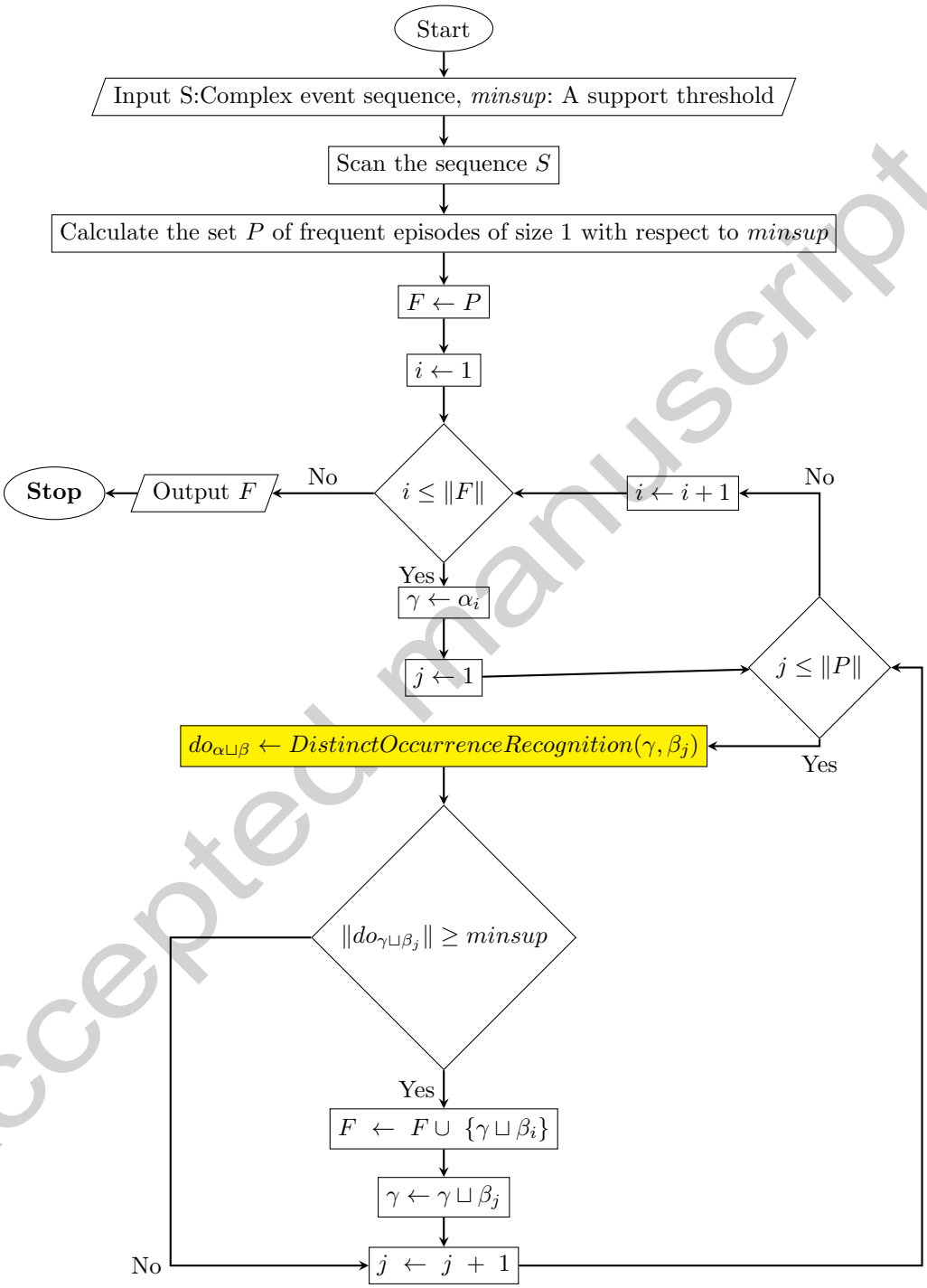
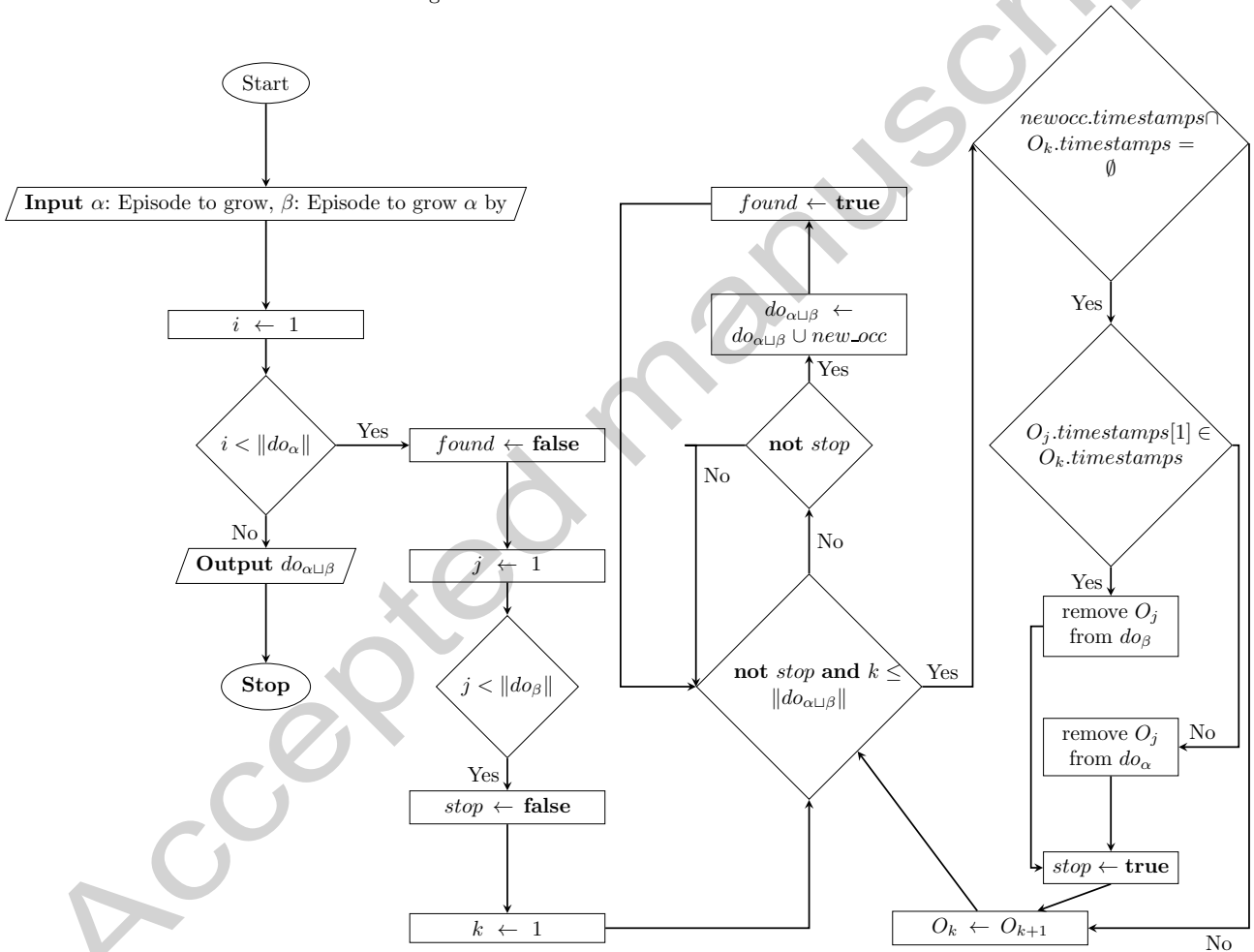


Fig. 12 Frequent Episode Generation function flowchart

Fig. 13 Maxmimal set of Distinct Occurrences Recognition function flowchart



References

[1] Kuo-Yu Huang, Chia-Hui Chang,Efficient mining of frequent episodes from complex sequences, Information Systems,Volume 33, Issue 1,2008,Pages 96-114,ISSN 0306-4379, https://doi.org/10.1016/j.is.2007.07.003.

[2] Wan, L., Chen, L., Zhang, C.: Mining Dependent Frequent Serial Episodes from Uncertain Sequence Data, In: Proc. IEEE 13th Int. Conf. on Data Mining, pp. 1211–1216. IEEE (2013)

[3] Wan, L., Chen, L., Zhang, C.: Mining frequent serial episodes over uncertain sequence data. In: Proceedings of the 16th International Conference on Extending Database Technology, March 2013 Pages 215–226https://doi.org/10.1145/2452376.2452403

[4] 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

[5] Achar, A., Laxman, S., Viswanathan, R., Sastry, P. S.: Discovering injective episodes with general partial orders. Data Min. Knowl. Discov. 25(1): 67-108 (2012)

[6] 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

[7] 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

[8] Mannila, H., Toivonen, H., Verkamo, I.: Dicovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery. **1**(3), 259–289 (1997)

[9] Iwanuma, K., Takano, Y., Nabeshima, H.: On anti-monotone frequency measures for extracting sequential patterns from a single very-long data sequence. In: Proc. 7th IEEE Conf. on Cybernetics and Intelligent Systems, pp. 213–217. IEEE (2004)

[10] Laxman, S., Sastry, P.S., Unnikrishnan, K.P.: Discovering frequent episodes and learning hidden markov models: A formal connection. IEEE Transactions on Knowledge and Data Engineering, **17**(11), 1505–1517

(2005)

[11] Mahesh, J., Karypis, G., Kumar, V.: A Universal formulation of sequential patterns. Technical report 99-021, University of Minnesota (1999)

[12] Achar, A., Laxman, S. & Sastry, P.S. A unified view of the apriori-based algorithms for frequent episode discovery. Knowl Inf Syst 31, 223–250 (2012). https://doi.org/10.1007/s10115-011-0408-2

[13] Zhou, W., Liu, H., Cheng, H.: Mining closed episodes from event sequences effi- ciently. In: Proceedings of PAKDD 2010, pp. 310–318 (2010)

[14] Huisheng, Z., Wang, P., Wang, W., Shi, B.: Discovering Frequent Closed Episodes from an event sequence, In: Proc. 2012 Int. Joint Conf. on Neural Networks (IJCNN'12), Brisbane, QLD (2012)

[15] Liao, G., Yang, X., Xie, S., Yu, P.S., Wan, C.: Two-Phase Mining for Frequent Closed Episodes. In: Proc. 16th Int. Conf. on Web-Age Information Management. LNCS, vol 9658, pp. 55–66, Springer (2016)

[16] Cheng-Wei Wu, Yu-Feng Lin ,Philip S. Yu,Vincent S. Tseng, Mining high utility episodes in complex event sequences, Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data miningAugust 2013 Pages 536–544https://doi.org/10.1145/2487575.2487654

[17] Fournier-Viger P., Yang P., Lin J.CW., Yun U. (2019) HUE-Span: Fast High Utility Episode Mining. In: Li J., Wang S., Qin S., Li X., Wang S. (eds) Advanced Data Mining and Applications. ADMA 2019. Lecture Notes in Computer Science, vol 11888. Springer, Cham. https://doi.org/10.1007/978-3-030-35231-8_12

[18] Fournier-Viger P., Yang Y., Yang P., Lin J.CW., Yun U. (2020) TKE: Mining Top-K Frequent Episodes. In: Fujita H., Fournier-Viger P., Ali M., Sasaki J. (eds) Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices. IEA/AIE 2020. Lecture Notes in Computer Science, vol 12144. Springer, Cham. https://doi.org/10.1007/978-3-030-55789-8_71

[19] 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

[20] Ao, X., Luo, P., Wang, J., Zhuang, F., He, Q.: Mining Precise-Positioning Episode Rules from Event Sequences. IEEE Transactions on Knowledge and Data Engineering, **30**(3), 530–543 (2018)

[21] Méger, N., Rigotti, C.: Constraint-based mining of episode rules and optimal window sizes, In: Proc. 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04), pp. 313–324 (2004)

[22] Liao, Guoqiong, Yang, Xiaoting, Xie, Sihong, Yu, Philip, Wan, Changxuan. (2018). Mining Weighted Frequent Closed Episodes over Multiple Sequences. Tehnicki Vjesnik.

[23] 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.

[24] 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

[25] 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.

[26] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, A fast algorithm for finding frequent episodes in event streams. In KDD, 2007.

[27] X. Ao, P. Luo, C. Li, F. Zhuang, Q. He. Online Frequent Episode Mining, In Proceeding ; ICDE Conference 2015

[28] 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.

[29] M. Ge, H. Bangui and B. Buhnova: Big Data for Internet of Things: A Survey. Future Gener. Comput. Syst. 87:601-614, 2018.

[30] N. A. Ghani, S. Hamid, I. A. T. Hashem and E. Ahmed. Social media big data analytics: A survey. Computers in Human Behavior, 101, 417-428, 2019.

[31] Olson, D. L., Auhoff, G. Association Rules. In: Descriptive Data Mining, Springer, pp. 67-76 2019.