People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Computer Science
Department of Computer Science

THESIS
In order to obtain the Doctorate degree in LMD (3^{rd} cycle)
Branch: Computer Science
Option: Networks and Multimedia

THEME

Fouille d'épisodes à partir de données incertaines (Episode mining from uncertain data)

By: Oualid Ouarem

Publicly defended on: 06/06/2024

In front of the jury composed of:

Pr. Messaoud Mostefai	University of B.B.A	**President**
Pr. Farid Nouioua	University of B.B.A	**Supervisor**
Pr. Philippe Fournier-Viger	Shenzhen University (China)	**Co-Supervisor**
Dr. Allaoua Hemmak	University of M'Sila	**Examiner**
Dr. Lyazid Toumi	University of Setif 1	**Examiner**
Dr. Abdelouahab Attia	University of B.B.A	**Examiner**

2024/2025

Dedication

To my parents

who taught me the patience

To my brothers and sister

who helped me during PhD period

To every one my teachers and colleagues

that encourage me to that last days

To the spirit of my brother, *Rabah Hammouche*

To every one

who didn't believe in our capabilities to finish this work

Oualid, OUAREM

Acknowledgment

I'd like to thank my supervisor Pr. **Nouioua Farid** for his advises and his
efforts to better improve this documents
I'd like also to thank Pr. **Philippe Fournier-Viger**
for his precious presence as a co-supervisor for this thesis.
His guides were very rich and very valuable.
A debt of gratitude is also owed
to Pr. **Messaoud Mostefai**, Dr. **Allaoua Hemmak**,
Dr. **Lyazid Toumi** and Dr **Attia Abdelouahab**
for taking time to read and evaluate this thesis.
Last but not least, my warmest gratitude goes to my family
especially my parents who were the source
of my patience and power to finish the thesis from the first year.

ملخص

يعتبر تنقيب البيانات عملية حيوية في اكتشاف المعرفة من البيانات. هدفها الرئيسي هو استخراج أنماط مثيرة للاهتمام تشير ضمناًا إلى علاقات هامة بين العناصر بشكل ضمني. تستخدم فروع تنقيب البيانات أنواع مختلفة من البيانات. يعتبر تنقيب الحلقات إحدى فروع تنقيب البيانات التي تهدف إلى الكشف عن معرفة قيمة من البيانات الزمنية في شكل سلسلة واحدة وطويلة من الأحداث. قد لا تحتوي السلسلة دائماًا على بيانات كاملة؛ قد تكون مضطربة أو تأتي من مصادر متعددة أو تكون جمعت بأخطاء. ونتيجة لذلك، هناك حاجة إلى تطوير وتصميم خوارزميات لاستخراج الحلقات المتكررة من البيانات غير المؤكدة. تقدم هذه المذكرة خوارزميات لأفكاراً جديدة بخصوص استخراج الحلقات المتكررة و العلاقات بين الحلقات في حالة البيانات المؤكدة، وتتناول أيضا التحديات المرتبطة بهذه المهام في سياق البيانات غير المؤكدة.

الكلمات المفتاحية: تنقيب الحلقات، قواعد الحلقات ، التنبؤ، معلومات غير المؤكدة

Abstract

Data mining is a critical process in the discovery of knowledge from data. Its primary objective is to extract interesting patterns that implicitly indicate significant relationships between items. Different branches of data mining manipulate various types of data. Episode mining is a subfield of data mining that aims to uncover valuable knowledge from temporal data in the form of a single, long sequence of events. The sequence may not always certain data; it may be noisy, sourced from multiple sources, or collected with errors. Consequently, there is a need to develop and design algorithms to extract frequent episodes from uncertain data. This thesis proposes novel algorithms for frequent episode and episode rule mining in the case of certain data and addresses also the challenges associated with these tasks in the context of uncertain data.

Key words Episode mining, episode rules, NONEPI, EMDO, UEMDO, prediction, uncertain data

Résumé

La fouille de données est un processus très important de la découverte de connaissances à partir de données. Il sert à découvrir des motifs importants qui peuvent révéler des relations significatives entre les données. Elle englobe des domaines différents d'où chaque domaine utilise un type spécifique de bases de données. La fouille d'épisodes est un domaine de la fouille de données qui sert à chercher des motifs appelés épisodes à partir des données temporelles sous forme d'une longue séquence d'événements. La majorité des algorithmes existants suppose que la séquence est toujours certaine. Cependant, la séquence n'est pas toujours certaine ; elle peut être bruyante, collectée à partir de plusieurs sources ou bien elle a été collectée avec des erreurs. Par conséquent, il est nécessaire de développer et de concevoir des algorithmes capables d'extraire des épisodes fréquents à partir de données incertaines. Cette thèse propose de nouveaux algorithmes pour la fouille d'épisodes fréquents et de règles d'épisode dans le cas de données certaines et aborde également le défi associé à ces tâches dans le contextes de données incertaines.

Mots clés Fouille d'épisodes, règles d'épisodes, NONEPI, EMDO, UEMDO, prédiction, données incertaines.

Table of contents

Abbreviations list

conf Confidence.

EMDO Episode Mining based on Distinct Occurrences.

EMDO-P Episode rules mining under distinct occurrences with pruning.

FEM Frequent Episode Mining.

FIM Frequent Episode Mining.

.

HUEM High-utility episode mining.

HUPM High-utility pattern mining.

Maxwin Maximum Window.

MINEPI MINimal occurrence-based frequent EPIsodes.

NONEPI NON-overlapping frequent EPIsode.

relSup relative support.

SDB Sequence database.

Sup Support.

UEMDO Uncertain Episode Mining based on Distinct Occurrences.

UEMDO-P Uncertain episode rules mining under distinct occurrences with pruning.

WINEPI WINdow-based EPIsode mining.

List of Figures

List of Tables

Chapter 1

General Introduction

The explosive growth of stored data has generated an emergent need for new techniques and automated tools that can assist in transforming the vast amount of data into useful knowledge. This led to the generation of a new frontier called *dataminig* and its subfields. Data mining is the task of extracting patterns representing knowledge implicitly captured in large databases, such as event logs, the web, or even data streams.

1.1 General context

Existing data mining methods use specific types of data. For itemset mining, the data comprises a set of transactions called the *transaction database*. Itemset mining is the first subdomain of the data-mining field. Another collection of techniques uses another data format called the *sequence database*. Sequence mining is an important field of datamining that focuses on extracting interesting patterns called *frequent sequences* with respect to the occurrence of items in databases. If the analysis focuses on the temporal relationships between data items in a single sequence, the task is called *episode mining*. Frequent episode mining uses a single long sequence of items, called textitevents, as input data and looks for episodes i.e., sub-sequences of events that frequently appear in the sequence. This thesis is situated in this context and aims to explore new algorithms for efficient episode mining from both certain and uncertain sequences of events.

1.2 Objectives

In general, the task of episode mining is to find important patterns, called *frequent episodes*, from the event sequences. Additionally, many techniques use such algorithms to derive more meaningful patterns called *episode rules*. Episode rules explain the correlations among events in the sequence, to be used later in future actions prediction in a given system. However, the majority of the proposed techniques assume that the data do not have any noise or imperfection i.e: the events are captured with absolute certainty of existence in the sequence. Unfortunately, in many scenarios, systems can record valuable data but faulty sensors, errors, or network transmission problems. Hence, the obtained data may be uncertain. Our objective in this thesis is twofold: First we propose new algorithms for mining frequent episodes and episode rules from certain data. The proposed algorithms in this context explore new frequency definitions (non overlapping and distinct) to propose new forms of episode rules that are suitable in practice for prediction tasks. The second objective is to extend the previous algorithms, namely those based on distinct occurrences frequency, to the case of uncertain data where uncertainty is expressed in the probabilistic framework.

1.3 Contributions

Our contributions in this thesis can be summarized in the four following points :

- An up-to-date state-of-the-art that presents and discusses the main existing algorithms in episode mining as well as their applications.
- A novel algorithm, called NonEpi, for mining frequent episode and episode rules from certain simple sequences where episode frequency is based on their non-overlapping occurrences.
- A novel algorithm, called EMDO, for mining frequent episode and episode rules from certain complex sequences where episode frequency is based on their distinct occurrences.
- To address the problem of data uncertainty, we proposed an extension of EMDO algorithm, called UEMDO, to discover frequent episodes and episode rules from complex probabilistic sequences.

1.4 Thesis structure

In addition to a general introduction and a general conclusion, this thesis is structured on five chapters organized as follows:
In **Chapter 2**, we start by giving a brief recall about data Mining and the general process of *Knowledge Discovery from Data*. Then we focus on pattern mining which is an important and active field of data mining. The chapter contains a brief introduction to different kinds of patterns to discover from different kinds of databases.

In **Chapter 3**, we discuss the latest state-of-the-art of frequent episode mining framework. The chapter provides an introduction to the field of pattern discovery from temporal data. It considers the definition of key concepts of episode mining technique as well as the variants of existing algorithms and their limitations. Recent applications that use episode mining approaches as well as current research opportunities are also discussed.

Chapter 4 is devoted to the presentation of our first contribution which is a novel approach for frequent episodes and episode rules mining based on non-overlapping occurrence-based frequency definition. The new approach proposes an important algorithm (called NonEpi) for prediction tasks owing to the new form of episode rules that has been proposed in that work.

In **Chapter 5**, we show the details of another novel method for episode rules discovery under a new frequency called *distinct occurrence-based frequency*. This chapter describes new algorithms called EMDO and EMDO-P for episode and episode rules discovery respectively.

Chapter 6 focuses on frequent episode and episode rule mining from uncertain data. We propose in this chapter an extension of EMDO and EMPO-P algorithms to extract expected frequent episode and adapted forms of episode rules from a sequence of probabilistic events.

Notice that for each of the three previous contributions, an extensive experimental study has been conducted on several datasets to show the efficiency or the proposed algorithms and their applicability in practical contexts.

Chapter 2

Introduction to Pattern Mining

Data is the fuel of the last decades, due to various real-life applications that manage our data either in our computer networks, the World Wide Web, or data stored every day from business or other domains. However, owing to the explosive growth of data, analyzing and covering useful information has become a challenging task. Consequently, there exists a collection of techniques and concepts called data mining designed to simplify the task of the analysis and understanding the relationships between items that constitute every piece of data. In this chapter, we explain the process of knowledge discovery from databases or **KDD**. Next, we focus on pattern mining, a subfield of data mining which aims at discovering relevant patterns and rules from large databases of different kinds.

2.1 The process of Knowledge Discovery from Data (KDD)

The whole process that aims at discovering and revealing the implicit and potentially useful knowledge embedded in databases is called *Knowledge Discovery from Databases*, or simply **KDD**. **KDD** is an iterative process that aims for extracting useful insights from databases (from simple databases to data warehouses). This process can be integrated into many fields, such as medicine, business, and biology. The process consists of several steps organized as follows:

1. **Data cleaning:** By the end of this step, the data won't contain any noise or inconsistencies. For instance, ignoring tuples that contain missing data, filling the missing data manually or using a global constant or a central tendency (such as the mean or the median) are techniques used to perform the cleaning sub-process like *Talend* and *Pentaho*.

2. **Data integration:** The aim of this step is to merge the data from multiple sources (spreadsheets, raw data, web, . . . , etc.) into one structure to reduce and avoid redundancy.

3. **Data Selection:** In this step, we select a relevant partition which is only dedicated to the specific wanted analysis.

4. **Data transformation:** In this preprocessing step, data is transformed to be suitable for the mining task. For instance, **aggregation** is an operation that may be performed to transform data, such as the *Group By* operation in the SQL language.

5. **Data-mining:** It is the heart of the KDD process in which efficient algorithms are applied to **mine** interesting knowledge that describe the hidden relationships between different attributes (items) from a large amount of data. According to the objective, different tasks may be achieved in this step including prediction, regression, classification clustering and pattern mining among others. In pattern mining context, the main task performed in this step is to extract all patterns depending on the type of database. For instance, if the database consists of transactions, the data-mining task here is to extract all *itemsets* from those transactions. If we have multiple sequences (for example, the sequences of customers' purchased items), the process will extract the important *sequences* and extract a special type of sequential pattern called *episodes* if we have only a single long sequence of events with their occurrence timestamps. In the next sections, we describe each variant in detail.

6. **Evaluation:** This is performed to identify truly interesting extracted knowledge. For example in pattern mining, this may involve the evaluation of the used thresholds such as *support* and/or *confidence* as measurement objectives.

7. **Knowledge presentation:** This is the final step of the KDD process where the extracted patterns are presented in a way that is understandable by users or experts.

2.2 Pattern mining

Now we focus on pattern mining which represents an important particular task of data mining where the objective is to extract relevant and useful patterns from large amount of data. Patterns can be directly extracted from raw data, used to understand data, and support deci-

sion making. Pattern mining algorithms have been designed to extract various types of patterns, each provides different information to the user and extracts patterns from different types of data. Popular patterns are sequential patterns and itemsets. The main task in FPM is to determine frequent patterns with a frequency that is not less than the support threshold. FPM techniques can be classified into several categories according to the database type. In this section, we present the main extensions of the FPM. Also, we present most known algorithms in the literature for each domain.

2.2.1 Frequent itemsets mining

Frequent Episode Mining (FIM)[1] is a popular framework designed in the aim of analyzing transaction databases. The first FIM algorithm was proposed by Aggarwal called association-rule [2]. It was first proposed for analyzing customer basket data and has recently been used in several applications.

2.2.1.1 Discovering frequent itemsets

Generally, the purpose of an FIM algorithm is to find a set of itemsets where **the support** exceeds a user-specified support threshold *minsup*. Let be a set of **items** $I = \{i_1, i_2, \ldots, i_m\}$. A set of transactions $D = \{T_1, T_2, \ldots, T_n\}$ such that each $T_j \subseteq I$ is a set of unique items. For instance, Table 2.1 shows an example of a transaction database where $I = \{a, b, c, e, f\}$ and $|D| = 2$. As mentioned before, the database in Table 2.1 may represent a specific customer's purchases. Hence, the items are products purchased by the customer.

Table 2.1: An example of transaction database

TID	Transaction
T_1	$\{a, e, b\}$
T_2	$\{a, c, f\}$

The **Support (Sup)** of an itemset is a measure of how often it occurs in the input database. It is also used to evaluate the itemset interestingness. Let D be a transaction database and X be an itemset, the support of X is formally defined as follows:

$$sup(X) = \frac{|\{T | X \subseteq T \wedge T \in D\}|}{|D|} \tag{2.1}$$

For instance, let $X = \{a, e\}$ be an itemset, and consider the example in Table 2.1. The support of X is equal to $\frac{1}{2}$ because it is only supported by transaction T_1. Consider a support threshold $minsup = 50\%$. the itemset X can be considered a frequent itemset because $sup(\{a, e\}) \geq minsup$.

Many algorithms have been proposed to retrieve frequent itemsets, such as associationrule [2] and Eclat[3]. Any algorithm may be designed to follow either a breadth-first or depth-first strategy.

2.2.1.2 Association rules generation

Some FIM approaches extend the original framework to reveal more meaningful patterns, called **Association Rules**. An **Association Rule** describes the relationship between two itemsets. Let be X_i and X_j two itemsets. **An association rule** is an implication $X_i \rightarrow X_j$. It is used to express the implicit relationship between any pair of itemsets (X_i, X_j) such that every transaction that contains X_i has a strong probability that X_j also exists in the same transaction T.

In addition to support, the **Confidence (conf)** is a measure to evaluate the interestingness of an association rule. The confidence of the rule $X_i \rightarrow X_j$ is the conditional probability $P(X_j|X_i)$. Formally, this is defined as follows.

$$conf(X_i \rightarrow X_j) = \frac{sup(X_i \cup X_j)}{sup(X_i)} = \frac{|\{T|(X_i \cup X_j) \subseteq T \wedge T \in D\}|}{|\{T|X_i \subseteq T \wedge T \in D\}|} \tag{2.2}$$

Given a support threshold $minsup$ and confidence threshold $minconf$. An association rule $X_1 \rightarrow X_2$ is said to be valid if and only if X_1 and X_2 are frequent according to $minsup$ and $conf(X_1 \rightarrow X_2) \geq minconf$.

2.2.2 Sequential pattern mining

Frequent itemset mining has become popular owing to its applications and community. However, if such an itemset mining algorithm is applied to data with time sensitivity or with a sequential order, the temporal or order information will be ignored; hence, it will fail to discover the strongest correlations in the data. Fortunately, sequential pattern mining has solved the problems of time and order in data.

2.2.2.1 Discovering frequent subsequences

Sequential Patterns Mining or **SPM** [4, 5, 6, 7, 8, 9], is another field of frequent pattern mining. It consists of discovering interesting sub-sequences from a set of sequences, where the interestigness of such a pattern can be evaluated using the occurrence frequency, length, or profit of the pattern.

State-of-the-art of Sequential pattern mining has encountered a huge amount of methods. Many algorithms have been proposed to enhance the existing algorithms or to propose new solutions to new challenges. The most well-known algorithms include GSP [8], PrefixSpam [9], CM-SPADE and CM-SPAM [7]..

The input data of any sequential pattern mining algorithm consist of sequences. A *sequence* is an ordered list of nominal values. a Sequence database (SDB) is a set of sequences in which each sequence has a unique identifier. Table 2.2 shows an example of an SDB. For example, it may constitute the purchased product items of four customers. For example, the first sequence represents the purchased items of the first customer, where he took in the first day two products together a and e then he took only the product f then the product g, and finally took b and d together.

Table 2.2: An example of sequence database

SID	Sequences
S_1	$\langle\{a,e\},\{f\},\{g\},\{b,d\}\rangle$
S_2	$\langle\{b\},\{f,a\}\rangle$
S_3	$\langle\{c\},\{e\},\{g,d\},\{b,e\}\rangle$
S_4	$\langle\{a,b\},\{d,c\},\{f,e\}\rangle$

Formally, given a set of items $I = \{i_1, i_2, \ldots, i_n\}$. A sequence $s = \langle X_1, X_2, \ldots, X_m\rangle$ is a set of itemsets such that $X_i \subseteq I$ for all $1 \leq i \leq m$. We denote by $k = \sum_{i=1}^{m} |X_i|$ the length of the sequence s. A sequence database is a set of sequences $SDB = \langle s_1, s_2, \ldots, s_n\rangle$ such that each sequence has a unique identifier.

A sequence $s_1 = \langle X_1, X_2, \ldots, X_n\rangle$ is said to be *contained* in another sequence $s_2 = \langle Y_1, Y_2, \ldots, Y_m\rangle$ if and only there exist integers $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ such that $X_1 \subseteq Y_{i_1}$, $X_2 \subseteq Y_{i_2}$, ..., $X_n \subseteq Y_{i_n}$. If a sequence s_i is contained in another sequence s_j, s_i is called *subsequence* of s_j and is denoted by $s_i \sqsubseteq s_j$.

The goal of sequential pattern mining is to find the subsequences of interest among a set of sequences in the SDB. To evaluate the interestingness of a pattern in sequential data, the support measure is used. The *support* of a sequence s in sequential pattern mining is the number of sequences in the SDB that contain the sequence s. Some sequential pattern mining algorithms use absolute support (see the definition in equation 2.3) to evaluate the interestingness of a pattern. Other algorithms use the relative support (relSup) denoted by equation 2.4 [4]

$$sup(s) = |\{s_i | s_i \in SDB \wedge s \sqsubseteq s_i\}| \tag{2.3}$$

$$relSup(s) = \frac{sup(s)}{|SDB|} \tag{2.4}$$

For example, consider the sequence database in Table 2.2. Let $s_\alpha = \langle\{a\}\rangle$ and $s_\beta = \langle\{a\}\{d\}\rangle$. Sequence s_α is included in sequences S_1, S_2 and S_3; hence, the absolute support of s_α is $sup(s_\alpha) = 3$. The absolute support of s_β is 2 because $s_\beta \sqsubseteq S_1$ and $s_\beta \sqsubseteq S_2$. The relative support of s_α and s_β is $\frac{3}{4}$ and $\frac{2}{4}$ respectively. Note that the anti-monotonicity property holds for the support count in sequence mining. This property states that the support of such a sequence does not exceed that of its subsequence. To prove this, consider the sequence $s_\gamma = \langle\{d\}\rangle$. It is easy to see that $sup(s_\beta) < sup(s_\alpha)$ and $sup(s_\beta) \leq sup(s_\gamma)$ with $s_\gamma \sqsubseteq s_\beta$ and $s_\alpha \sqsubseteq s_\beta$. Given a support threshold *minsup*. SPM aims to find all sequences with a support value not less *minsup*. Many algorithms exist that aim to efficiently extract all the frequent sequences from a sequence database. Each algorithm uses a proper database representation (vertical or horizontal) and prunes the search space. It also uses a strategy for exploring the search space: breadth-first algorithms that perform candidate generation or depth-first search algorithms.

2.2.2.2 Sequential rules mining

An important extension of sequential pattern mining is the extraction of relationships between subsequences from multiple sequences. This variant is known as sequential rule mining. A sequential rule is an expression of the form $X \rightarrow Y$ indicating that if some items X appear in a sequence, it will be followed by other items Y with a given confidence. This concept is similar to that of the association rule, as mentioned previously, except that X must appear before Y. The confidence of a sequential rule is an important measure for evaluating its strength. For example, consider the sequence of Table 2.2. Let $s_\alpha = \langle\{a\}\rangle$ and $s_\beta = \langle\{b\}\rangle$. It is easy to see that every

sequence that contains s_α also contains sequence s_β which indicates that whenever a sequence contains itemset $\{a\}$, there is a strong probability that it is followed by a sequence that contains itemset $\{b\}$. The relationship between two sequences is called a *sequential rule* $\langle\{a\}\rangle \rightarrow \langle\{b\}\rangle$.

Given a sequence database, a *support* threshold *minsup* and a *confidence* threshold *minconf*. The problem of a sequential rule mining algorithm is to find all frequently valid *sequential rules* such that the support is not less than *minsup* and the confidence is not less than *minconf*. Many algorithms have been proposed for mining sequential rules like RuleGrowth [10] and ERMinner [11].

2.2.3 Frequent episode mining

Pattern mining also encounters another extension, called Frequent Episode Mining (FEM) [12, 13, 14]. Episode mining is a type of sequential pattern mining. The main difference from sequential pattern mining is that it aims to find subsequences called *episodes* from a *single long sequence* of events, which was first introduced by Mannila et al. [12] in 1997. The first two algorithms called WINdow-based EPIsode mining (WINEPI) and MINimal occurrence-based frequent EPIsodes (MINEPI). Many recent works extend the original one to a huge number of applications, such as cybersecurity and stock trading. All algorithms in the FEM focus on three concepts, defined as follows:

Let $\varepsilon = \{e_1, e_2, \ldots, e_k\}$ be a set of event types. An **event** is a pair (e,t) where $e \in \varepsilon$ is a nominal value that represents the event type, such that $e \in \varepsilon$ where ε is the set of all event types and $t \in \mathbb{N}$ is the occurrence timestamp of the event in the sequence.

An **event sequence** is $S = \langle(e_1,t_1),(e_2,t_2),\ldots,(e_n,t_n)\rangle$ is an ordered set of events (e_i,t_i) such that $e_i \in \varepsilon$ and $t_i \in \mathbb{N}$ is its occurrence time in sequence S and for all $i, j \in \mathbb{N}$ if $i < j$ we have $t_i < t_j$. The third concept is called **episode**. An episode is a set of nodes, where each node is associated with an event type in the episode. The task of such an episode mining algorithm is to find all **frequent episodes** with respect to the support threshold, such that the support of the episode exceeds the support threshold. Any algorithm in episode mining maintains its proper support definition according to how an episode occurs in the sequence to be considered as a valid episode's occurrence. In the next chapters, we will show all concepts of episode mining in detail, as well as our contributions to the field of frequent episode mining.

2.3 Pattern mining with concise representations

The increasing accumulation of datamining has prompted the adoption of new, compact patterns with efficient representations. These representations are employed to effectively extract valuable insights from data at a minimal cost and without loss of information. There are three primary categories of concise representation:

2.3.1 Closed frequent patterns

A closed pattern is defined as a pattern that is both frequent and does not have a super-set with the same support count. More formally, the set of closed patterns, denoted as **CP**, can be expressed as follows.

$$\mathbf{CP} = \{p \mid p \in \mathbf{FP} \wedge \nexists p' \in \mathbf{FP}\ s.t.\ p \subset p' \wedge sup(p) = sup(p')\}$$

Closed frequent pattern recognition is utilized by a variety of datamining algorithms to minimize redundant patterns and enhance the mining efficiency [15, 16, 17, 18, 19].

2.3.2 Maximal frequent patterns

Maximal patterns represent are frequent patterns characterized by their absence of frequent super-sets. To provide a precise formal definition, the set of maximal frequent patterns is outlined as follows:

$$\mathbf{MP} = \{p \mid p \in \mathbf{FP} \wedge \nexists p' \in \mathbf{FP}\ s.t.\ p \subset p'\}$$

Maximal patterns have been used to extract frequent maximal itemsets [20], frequent maximal sequential patterns [21] and maximal frequent episodes [22]. The set of maximal patterns is more compact than the set of closed patterns. In other words, the length of the set **MP** is less than the number of **CP** which does not exceed the number of all frequent patterns **FP**.

2.3.3 Generator patterns

Another specific type of patterns is called generators [23, 24]. A pattern is said to be a *generator* if it has no subset with the same support count. Formally, a set of generator patterns

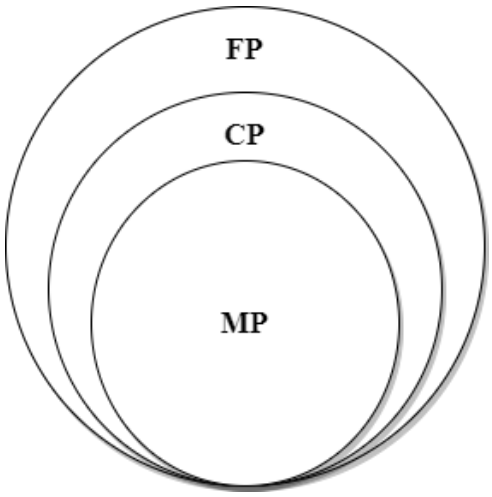


Figure 2.1: Relation between Frequent Patterns, Closed patterns and maximal patterns

is defined as follows:

$$\mathbf{GP} = \{p \mid p \in \mathbf{FP} \wedge \nexists p' \in \mathbf{FP}\ s.t.\ p' \subset p \wedge sup(p') = sup(p)\}$$

2.4 Beyond frequent pattern mining

Frequent Pattern mining is based on the assumption that the frequent patterns are interesting. However, this assumption does not hold in many applications. On the one hand, for some applications, a pattern may be frequent but may be uninteresting, as it describes common behavior, and hence, it may yield a low weight. However, there may exist other patterns that are infrequent but yield a higher weight. Fortunately, a recent study called *high-utility pattern mining* [25] was proposed to address the limitation of the pattern weight. However, for some applications, a pattern may be frequent, but its occurrence is irregular; hence, it is considered an uninteresting pattern. Based on this hypothesis, an approach called *periodic pattern mining* is proposed to extract patterns with regular occurrences, particularly for databases with time information.

2.4.1 High Utility pattern mining

The goal of High-utility pattern mining (HUPM), is to find patterns with a greater profit. HUPM has various pattern-based extensions. In general, finding patterns with utility over a user-defined utility threshold is the goal of HUPM. A list of some high-utility pattern algo-

rithms is shown in Table 2.3. The input of such a HUPM algorithm is the database (transaction databases, sequence databases, or a single long event sequence) with an external utility table that associates, for each item, a positive integer as an importance degree in the database, and a user-defined utility threshold.

Table 2.3: High-utility pattern mining algorithms

High-utility itemsets mining	High-utility sequential pattern mining	High-utility episode mining
IHUP[26], HUP-Growth [27], HUI-Miner [28]	UL,US [29], UM-Span [30], USpan [31], PHUS [32]	UP-Span [33], TSpan [34], HUE-Span [35]

2.4.2 Periodic pattern mining

Periodic-frequent pattern mining was first introduced by Tanbeer et al. [36]. It aims to identify patterns that occur periodically in the databases. A frequent pattern is said to be periodic if, and only if, it occurs frequently at regular time intervals. Formally, any periodic pattern satisfies the following criteria: (*i*) it is frequent with respect to a user-defined support threshold and (*ii*) its periodicity must not be greater than the maximal periodicity value. Periodic pattern mining has not been studied extensively as traditional frameworks [37, 38, 39, 40]

2.5 Conclusion

In this chapter, we introduced the field of pattern mining. We started by briefly defining the **KDD** process and then explained the domain of pattern mining and its variants and extensions seen as a particular data mining task. In the next chapter, we will discuss in more detail the state-of-art of one pattern mining sub-field called *episode mining* which is our research axis in the present thesis.

Chapter 3

Episode Mining: State-of-the-Art

Episode mining is a sub-domain of pattern mining, focusing on the discovery of interesting episodes, which are essentially sub-sequences of events within a given event sequence. A primary task in this context is that of FEM, wherein the goal is to identify episodes that occur frequently in a sequence of events. This challenge has been extended in diverse ways to cater to specific requirements. Research has demonstrated that episode mining can unveil valuable patterns applicable to diverse areas such as web stream analysis, network fault management and cybersecurity among others. Moreover, frequent episodes and episode rules prove beneficial for predictive purposes.

As an actively evolving research field, episode mining has witnessed significant progress over the past 25 years. This chapter serves as an introduction to episode mining, offering insights into recent developments and highlighting research opportunities. Additionally, we will explain the key concepts in this field.

3.1 Introduction

Data mining techniques are employed to identify and illustrate implicit relationships within extensive datasets. These relationships can be clarified for users and effectively managed by various agents within intelligent systems. Pattern mining algorithms can analyze various types of data, including transactions, sequences, time series, and graphs. One common real-life data type is event sequences, which represent an ordered list of events. In an event sequence, each

event is associated with a time occurrence (referred to here as occurrence for simplicity). Examples of such sequences include a web click stream, a tourist's sequence of points of interest visited in a city or an event log from a complex system[41].

In specific scenarios, it is necessary to perform sequential analysis for the examination of a sequence database, which mainly comprises a set of sequences. To address this need, the domain of sequential pattern mining (SPM) has been introduced. SPM, a sub-field of pattern mining, focuses on discovering sub-sequences in data that unveil significant relationships between elements [4]. The support of a sub-sequence is determined by the number of sequences in which it is present. Detecting recurring sequential patterns can be beneficial, for instance for uncovering common shopping patterns among multiple customers in a market dataset or for recognizing word sequences that occur frequently across multiple sentences in a text. A significant variation of SPM is sequential rule mining, where the objective is to discover rules expressed as $L \rightarrow R$ with high confidence (conditional probability) in a database of sequences [5, 6].

Some applications necessitate the analysis of data characterized by long event sequence to unveil patterns or construct models that aid in comprehending the past or predicting the future. To address this requirement, the concept of episode mining was introduced, encompassing a set of techniques tailored for this purpose. These techniques represent descriptive data mining methods utilized to identify compelling relationships among events within event sequences. Such relationships materialize as episodes, i.e., specific subsequences of events deemed interesting based on criteria like their frequency of occurrence. Moreover, there are implication-style rules referred to as episode rules that can reveal robust correlations between sub-sequences of events. The field of episode mining has garnered significant attention from both researchers and practitioners, in part due to the interpretability of the information offered by episodes.

For instance, An episode rule detected in alarms log of a telecommunication network suggests that it is typical to activate an alarm A in one device following the occurrence of alarm B in another device. These regulations provide insight into comprehending the interconnections among alarms and can be utilized to refine network monitoring and maintenance. This information permits for more efficient network management by prioritizing attention on the most critical alarms [42].

Episode mining [43] has been employed to derive patterns from sequential data in numerous real-world applications, like analyzing telecommunication network data [12], discovering and preventing attacks [44], examining financial occurrences and patterns in stock movements [45], analyzing web logs, [41], detecting intrusions [46], identifying Internet worms [47], root-cause analysis of machine faults [48] and detecting anomalies[49].

Algorithms of episode started in 1997 when Mannila et al. [12, 50] defined the principal task of episode mining: identifying frequent episodes. In the context of FEM, when a long sequence of events is presented, the objective is to discover sub-sequences that occur with sufficient frequency, meaning that the number of occurrences in the input sequence exceeds a specified threshold known as the minimum support (denoted *minsup*). These identified sub-sequences are termed **frequent episodes.** To illustrate that, consider an event sequence obtained from the log of a server database, frequent episodes disclose patterns of service queries that occur frequently, providing opportunities for optimization to enhance overall performance.

Previous research extensively investigated three main types of episodes: *serial episodes* when the events are totally ordered, *parallel episodes* when the events may appear in any order, [14, 12, 50] and injective episodes with an unrestricted partial order [51]. In addition to the aforementioned fundamental episode types, the domain of episode mining has witnessed the proposition of several extensions. Notably, efforts have been made to address the challenge of information overload for users by exploring subsets of episodes with remarkable characteristics. Examples of such subsets include frequent closed episodes [22] and maximal episodes [52]. These efforts aim to decrease the volume of presented episodes while retaining those deemed significant based on specific properties. Furthermore, supplementary constraints were introduced to refine the selection of episodes with enhanced significance, including constraints pertaining to the temporal intervals between successive events. [52].

Several algorithms have been proposed for the extraction of episodes within dynamic sequences, with a specific focus on identifying the top-k most significant episodes [53], as opposed to episodes exceeding a predefined frequency threshold. Another area of research involves the exploration of alternative frequency definitions for the identification of frequent episodes. Examples include window-based frequency [12, 50], non-overlapped frequency [54, 51, 55, 56], minimal occurrence-based frequency [57, 58, 50, 59], head frequency [60], and total frequency [13, 61] (for a comprehensive review of diverse functions assessing frequency,

refer to [33]).

Moreover, some researches have delved into alternative criteria for the selection of interesting episodes, like the consideration of their utility or importance. Additionally, evaluative functions have been explored, taking into consideration factors such as uncertainty and/or imprecise information.

Furthermore, numerous algorithms for episode rule mining have been disseminated, extending episode mining algorithms to deduce rules in the form of implications that clarify relationships among episodes [12, 62, 63, 64]. The practical applicability of episode rules extends to diverse tasks, including the prediction of future phenomena and the explication of specific occurrences grounded in past events.

3.2 Frequent episode mining framework

FEM is a widely adopted framework for analyzing temporal data [43]. It is employed across diverse domains that involve time-based information. The algorithms for frequent episode mining are designed to identify all recurring episodes, either from a simple sequence of events [50, 65, 66] or from a complex event sequence [67, 68]. In the former case, the sequence prohibits the simultaneous occurrences of events, while the latter allows events to happen simultaneously.

3.2.1 Preliminaries of FEM

The standard input for an FEM [69] algorithm includes a sequence of events and a user-defined frequency threshold known as *minsup*. This threshold specifies the required number of occurrences for a frequent episode.

Consider a event types set denoted as $E = \{E_1, E_2, ..., E_n\}$. A **sequence** of events is represented as a triplet $S = (s, T_s, T_e)$, wherein s denotes an events list recorded within the temporal interval from T_s to T_e.

To provide more clear explanation, the sequence s forms an ordered set of events, expressed as $s = \langle (I_1, T_1), (I_2, T_2), \ldots, (I_m, T_m) \rangle$, where $I_i \in E$ denotes an event type, and T_i corresponds to the timestamp of occurrence of the event ($1 \leq i \leq m$). The event pairs within s are arranged in

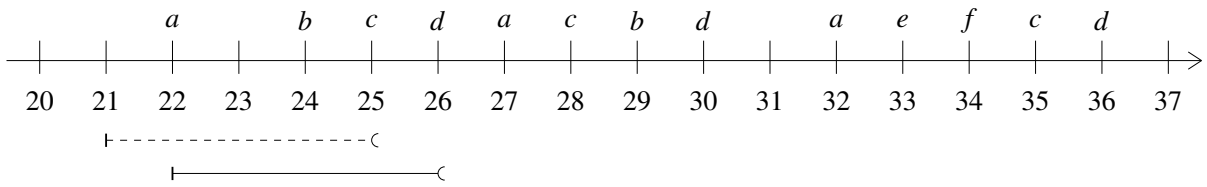


Figure 3.1: An example of event sequence

ascending order based on their timestamps, allowing for the potential occurrence of an event type multiple times within the sequence.

For instance, Figure 3.1 illustrates a sequence of 13 events recorded between times $T_s = 20$ and $T_e = 36$. This sequence might illustrate the event log of a database system, with event types $E = a, b, c, d, e, f$ representing disk operations like opening, closing, reading, or writing in a file. In this sequence, the recorded events are listed as: $s = \langle (a,22), (b,24), (c,25), (d,26), ..., (d,36) \rangle$. In the following subsections, we consider that the expressions *event* and *event type* are used mutually, provided the context is clear.

An **episode** is characterized as an events collection that occur jointly within a sequence. Formally, an episode is a triplet $\alpha = (V, \leq, f)$ is precisely delineated by a nodes set V, a (partial or total) order $\leq$ imposed on V, and a mapping function $f : V \to E$ that establishes associations between each node in V and a corresponding event type drawn from the set E.

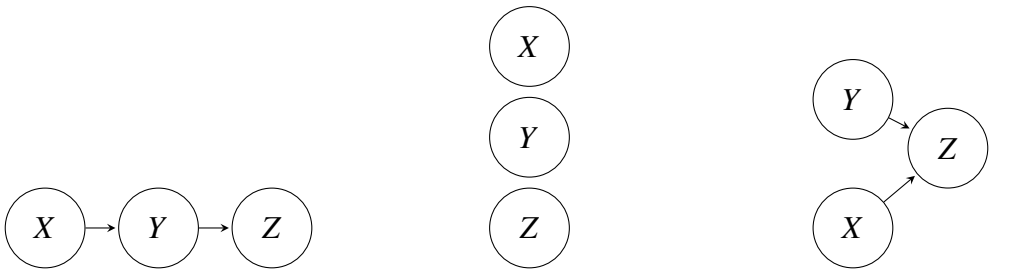
Two distinct types of episodes are commonly studied based on the provided order $\leq$:

1. If order $\leq$ is total, we refer to the episode α as a *serial episode*. It can be represented as $\alpha = A_1 \to A_2 \to \cdots \to A_k$, where $A_i \in E$ for $1 \leq i \leq k$, and α_i denotes the i^{th} event type in episode α.

2. If there are no constraints on the order between events, α is termed a *parallel episode* [70, 71], represented as $\alpha = A_1 A_2 \ldots A_k$. Composite episodes, which are a serial combination of parallel events, are also possible [52].

Moreover, the concept of episodes can be customized to match specific applications and data characteristics, including scenarios involving time-sensitive and uncertain data [72, 73]. Further discussions on these extensions will follow in subsequent sections.

For a given episode $\alpha = (V, \leq, g)$, its size, denoted as $|\alpha|$, represents the count of events it comprises, indicating the number of events it includes ($|\alpha| = |V|$). An episode α that includes



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

Figure 3.2: The three primary episode types

exactly k events is specifically referred to as a k-episode.

To illustrate, Figure 3.2 shows the primary three types of episodes. Each depicted episode within the illustration comprises a set of nodes denoted $V = \{v_1, v_2, v_3\}$. In this context, each node v_i is linked to an event type, that is: $f(v_1) = X$, $f(v_2) = Y$, and $f(v_3) = Z$. Figure 3.2(a) illustrates a serial episode such that the order is $\leq = \{(v_1, v_2), (v_2, v_3)\}$. The occurrences of the illustrated episode in Figure 3.2(a) in a sequence are considered if an event of type X is succeeded by an event of type Y, and then by an event of type Z. Figure 3.2 (b) depicts an example of a parallel episode with no clear order ($\leq = \emptyset$). Finally, Figure 3.2(c) illustrates a composite episode such that $\leq = \{(v_1, v_3), (v_2, v_3)\}$. This episode implies that events of types X and Y need to appear before the occurrence of event Z; however, no constraints are imposed on the sequential arrangement of events of types X and Y relative to each other.

An episode is deemed frequent within an event sequence when it manifests with a notable recurrence in the sequence. The quantification of an episode's frequency necessitates the adoption of a specific frequency definition, delineating the methodology for enumerating the occurrences of the episode within the event sequence. Given a sequence S with events set denoted as $s = \langle (I_1, T_1), (I_2, T_2), \ldots (I_m, T_m) \rangle$ and an episode $\alpha = (V_\alpha, \leq_\alpha, f_\alpha)$. An **occurrence** of α the sequence S is represented by a mapping $h : V_\alpha \to \{1, 2, \ldots, m\}$. This mapping satisfies the following condition: $\forall v \in V_\alpha$, $f_\alpha(v) = I_{h(v)} \wedge \forall w \in V_\alpha, v <_\alpha w$ we have $T_{h(v)} < T_{h(w)}$ [13].

For an example of this, let us examine the sequence shown in Figure 3.1. Let $\alpha = a \to b$ be a serial episode with $V_\alpha = \{v_1, v_2\}$ and $f_\alpha(v_1) = a$, $f_\alpha(v_2) = b$. α is present in the events (a, 22),(b, 24) because, for $h(v_1) = 22$ and $h(v_2) = 24$, $f_\alpha(v_1) = I_{h(v_1)} = a$ and $f_\alpha(v_2) = I_{h(v_2)} = b$, respectively. Additionally, the episode's events maintain their chronological sequence, meaning that because $v_1 <_\alpha v_2$, we get $T_{h(v_1)} = 22 < T_{h(v_2)} = 24$. It is clear that the occurrences of (b,29),

(a,32) are not instances of α. As an additional example, consider a parallel episode $\beta = bc$. There are two instances of β: $(b,24),(c,25)$ and $(b,29),(c,28)$.

The notions of sub-episode and super-episode play a fundamental role in the field of episode mining, representing distinct types of dependencies among couples of episodes having common events (nodes). Generally, consider two episodes γ and Δ, the consideration of γ as a sub-episode of Δ (and vice versa, Δ as a super-episode of γ) depends upon the nodes of α forming a subset of the nodes of Δ, with their respective order in γ being consistent with that in Δ. The formal expression for this connection is $\gamma \sqsubseteq \Delta$. As an illustrative example, a serial episode $\gamma = a \rightarrow c$ is considered as a sub-episode of another episode $\beta = a \rightarrow b \rightarrow c$, thereby satisfying $\alpha \sqsubseteq \Delta$. In contrast, a parallel episode $\theta = bc$ serves as a sub-episode of Δ but not of γ.

Note that numerous methods adapt the previously mentioned definitions to adapt the mining of episodes to specific domains or to accommodate particular data types. As an example, the *suffix* and *prefix* are specific kinds of a super-episode, and they are included into the NON-overlapping frequent EPIsode (NONEPI) algorithm, which is described in [43]. Similarly, the 2PEM method (described in [15]) looks at three types of sub-episodes: **forward**, **middle**, and **backward** extension (in the context of closed episode mining), which will be discussed later.

3.2.2 Definition of frequency in episode mining

As was previously established, an episode's frequency, often referred to as **support**, measures how frequently an episode occurs within a sequence of events. Various definitions of episode frequency exist, potentially resulting in different calculations. Broadly speaking, methods for calculating episode frequency can be classified into two categories depending on whether a sliding window is used to count occurrences or not:

- **Window-based frequency definitions.** They use a window with fixed width Maximum Window (Maxwin) to record occurrences of every given episode in the sequence. The number of windows of size k that include at least one occurrence of the given episode is considered to determine the frequency of that episode. Examples of window-based frequency metrics include windows-based frequency as detailed in [12], head frequency as discussed in [74], total frequency as presented in [74], and non-interleaved frequency as elaborated in [75].

- **Occurrence-based frequency definitions.** These definitions precisely monitor the episodes' occurrences across the sequence, without initially subdividing the sequence into subsequences, except in cases where the user specifies conditions on the occurrences' size, commonly referred to as the *span* or *gap* constraints [14]. This category of definitions encompasses metrics such as minimal occurrence-based frequency as delineated in [50], non-overlapped occurrence-based frequency as discussed in [43, 67], and distinct occurrence-based frequency as detailed in [13].

Table 3.1 offers a comprehensive summary of widely used frequency definitions, categorized into two groups, with additional details provided. Every definition is accompanied by its type and specifies the representative episode mining algorithm(s) that utilize it. The table further indicates whether a definition fulfills the anti-monotony property. This property is essential for the design of efficient algorithms for frequent episode mining, as it ensures that an episode's frequency should not surpass that of its sub-episodes. This topic will be further explored in the following subsection.

Table 3.1: Frequency definitions overview

Type	Name	Monotonic	Representative algorithm(s)
Window-based	window-based frequency	Yes	WINEPI [12], EpiBF [76], WinMiner [62]
	head frequency	No	EMMA [60]
	total frequency	Yes	FEM-DFS [65]
	non-interleaved frequency	No	NOE-WinMiner [75]
Occurrence-based	minimal occurrence-based	No	MINEPI [12], MEELO [77], PartiteCD[78]
	non-overlapped occurrence-based	Yes	NONEPI[43], POERM [67]
	distinct occurrence-based	Yes	ONCE+ [79]

The constraints associated with window-based frequencies encompass the possibility of multiple enumerations of a given episode occurrence in the event of its appearance across distinct windows. For example, Figure 3.1 shows a sequence with two window times, $f_1 = (\langle(a,22),(b,24)\rangle,21,25)$ (the dashed line) and $f_2 = (\langle(a,22),(b,24),(c,25)\rangle,22,26)$ (the straight line), where the windows times both include a succeeded by b. This could result in a double count of the serial episode indicating that a is followed by b. Another limitation is that the fre-

quency of an episode can vary depending on the width of the window. For instance, an episode may be infrequent for one window width, but frequent for another. In order to discover the ideal window width, the user may need to do several experiments, which might take significant time. Alternatively, they can depend on other methods to determine the optimal width. Fortunately, contemporary algorithms, exemplified by WinMiner, have overcame this issue [62].

Numerous FEM algorithms are formulated to identify episodes characterized by frequent occurrences within a given sequence, employing either a window-based or occurrence-based frequency definition. Yet, within an occurrence-based frequency framework, extended episodes may exhibit high frequency despite containing events with substantial temporal separation. In contrast, a window-based definition guarantees proximity of events but may overlook larger episodes exceeding the stipulated window size. To address these limitations, alternative definitions and algorithms have been proposed, such as the so-called DFS depth-first search algorithm[80] proposed by Cule et al. They established two parameters, *cohesion* and *coverage*, to capture the interest of a given episode, as follows:

- *The "coverage"* metric computes the frequency with which an event within an episode appears in a sequence denoted as S, it is computed as follows:

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

  Here, α represents a specified episode, and $N(\alpha)$ signifies the collection of all instances involving its constituent events. The term $P(\alpha)$ denotes the coverage of the episode $|\alpha|$ within the sequence S.

  Cohesion is an measurement of the proximity between events forming an episode when manifested in a sequence. Its calculation is delineated as follows:

- The *cohesion* of an episode is determined by how close together its occurrences are when they happen in a particular order. it is computed as follows:

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

  Here, $\overline{W}(\alpha)$ represents the average length of the shortest intervals, denoted as $W(\alpha,t)$,

that encompass the episode α. It is calculated as follows:

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

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 computes the degree of interest associated with an episode through the following procedure:

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

As a result, with a designated threshold for interestingness denoted as min_{int}, an episode α is deemed interesting if and only if $I(\alpha) \geq min_int$. Employing a Depth-First Search (DFS) approach, it has been asserted that the generated output encompasses cohesive episodes featuring events in close temporal proximity [80]. Note that alternative definitions may be formulated to address specific requirements.

In the next subsection, we will examine two strategies for exploring the search space in the identification of frequent episodes: namely, breadth-first search and depth-first search. Next, we will provide a summary of the principal methods employed by the widely recognized frequent episode mining (FEM) algorithms.

3.3 Search strategies for FEM

Since Mannila introduced the problem of frequent episode mining [12], several algorithms have been developed to improve the FEM process. FEM algorithms utilize distinct data structures and are crafted to accommodate multiple frequency definitions. In general, FEM algorithms can be categorized with respect to their search strategy, which is either a breadth-first search or a depth-first search:

1. **Breadth-first algorithms:** Commonly referred to as level-wise algorithms. They process in the exploration of episodes through a cyclic progression between two fundamental stages: (1) candidate generation and (2) frequency validation. In the first phase, candi-

date episodes are derived from smaller episodes, and in the second phase, the frequency of these candidate episodes is assessed to ascertain their status as frequent occurrences. Breadth-first search methodologies, guided by the anti-monotonicity property, prioritize the enumeration of lengthier episodes while disregarding numerous infrequent episodes. Illustratively, the CloEpi algorithm [75] initiates by comprehensively scanning the entire event sequence to identify frequent 1-episodes. Subsequently, CloEpi conducts a re-examination of the sequence, systematically identifying frequent 2-episodes, 3-episodes, and so on, up to m-episodes, such that m indicates the width of the episode. There exists several other level-wise algorithms, namely WINEPI, MINEPI [12], and FEM-BFS [14, 75].

2. **Depth-first search algorithms:** Methods like MANEPI [55] and FCEMinner [17] conduct a single pass over the event sequence to extract all 1-episodes. Subsequently, these algorithms endeavor to iteratively expand each episode by augmenting it with a frequent event, leveraging the anti-monotonicity property for the purpose of search space reduction. The adoption of a depth-first strategy typically results in decreased time and memory requirements during the process of frequent episode mining.

In the aim of formulating an efficient algorithm for FEM, a critical imperative involves circumventing the exhaustive exploration of the complete search space of episodes within the input sequence to identify frequent episodes. This is particularly pivotal because of the potential enormity of the search space for comparatively extensive sequences, which leads to protracted runtimes and heightened memory requirements. Using the frequency measure's anti-monotony quality is a key method for overcoming this challenge. Let $sup(\gamma)$ represent the frequency of an episode γ. This property states that for every couple of episodes γ and δ where $\gamma \sqsubseteq \delta$ (γ is a sub-episode of δ), the relationship $sup(\gamma) \geq sup(\delta)$ holds. Consequently, if an episode α infrequency, all its super-episodes are inherently infrequent, obviating the need for exploration and effectively reducing the amount of time and memory required. For instance, we outline the principal procedures of two widely adopted and exemplary FEM algorithms, namely, WINEPI and FEM-DFS, which employ breadth-first and depth-first search strategies, respectively.

3.3.1 WINEPI : Window-based episode mining method

WINEPI is the first Frequent Episode Mining method designed for discovering interesting (frequent) episodes [12]. It has been employed in various studies, including the identification and prevention of attack episodes [44] and the analysis of logs of mobile payment [81]. The input of WINEPI algorithm consists of a single long events sequence, denoted as S with respect to the format discussed in the previous section. The algorithm also takes an event types set, denoted by E, a specified sliding window size denoted as *Maxwin* and a minimum support threshold defined by the user referred to as *minsup* (equivalently recognized as the minimum frequency threshold). The set of frequent episodes inside the event sequence S, with regard to the sliding window size *maxwin* and the threshold values of *minsup*, is the output that WINEPI produces. This algorithms operates in two primary stages: (1) "candidate generation" and (2) "frequency check":

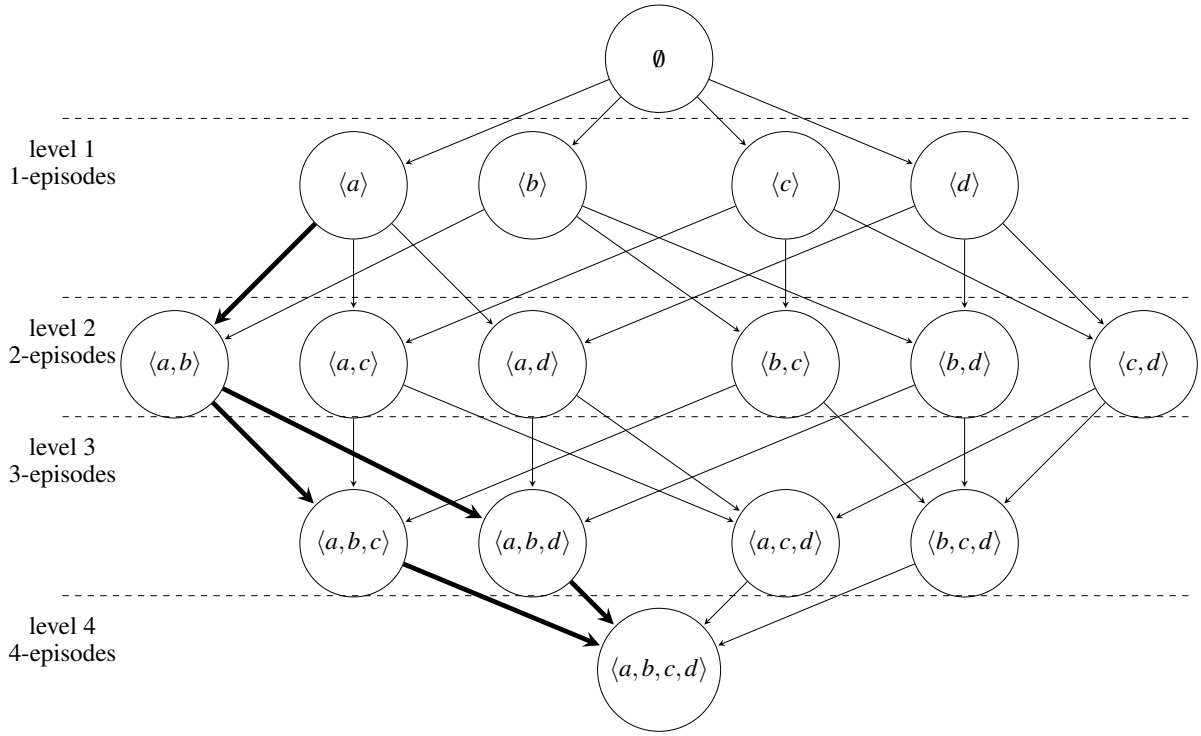


Figure 3.3: WINEPI's Breadth-First Search Process lattice representation

- *Candidate generation.* It involves generating candidate episodes, either parallel or serial, that might occur frequently. Figure 3.3 provides a visual representation of the enumeration process through a tree structure. The algorithm first establishes a list to record episodes of size $l = 1$, or 1-episodes, in which every event type is considered as an individual episode. Subsequently, a frequency check is conducted (discussed in the next

subsection) to performed to determine the set F_1 comprising frequent episodes of size $l = 1$ (i.e., 1-episodes). The set of candidate 2-episodes, C_2, is obtained by combining each pair of frequent episodes. It is represented by the set $C_2 = \{< a,b >, < a,c >, < a,d >, < b,c >, < b,d >, < c,d >\}$. The next set of iterations consists of exploring of 3-episodes and, subsequently, the identification of the 4-episode $< a,b,c,d >$(the biggest candidate episode that can be obtained from these event types).

- *Frequency check*. This stage involves quantifying the frequency of candidate episodes through the utilization of a sliding window mechanism applied to the sequence. In WINEPI, the frequency of a given episode α, is determined as the proportion between the windows count, denoted as w, of duration *win* containing more than or one occurrence of α, and the total windows number (represented as $|W|$).

  Note that there is a difference in tracking occurrences between serial and parallel episodes. For the tracking of serial episodes, WINEPI and analogous algorithms focusing on serial episodes, such as "DiscoveryNonOver" and "DiscoveryTotal" [14], "UVSEM" [13](**U**nified **V**iew for **S**erial **E**pisodes **M**ining), and WINEPI [12], maintain a Finite state automaton (FSA). The FSA designed for monitoring all instances of an episode $\alpha = A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_k$ encompasses $(k+1)$ states. The initial k states are denoted in the form (i, A_{i+1}),

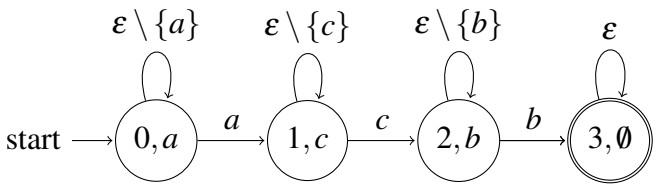


Figure 3.4: Automaton that tracks all occurrences of $\alpha = a \rightarrow c \rightarrow b$

meaning that the automaton has traversed the initial i events and anticipates the subsequent event A_{i+1}. The final state $(k, \emptyset)$ serves as the accepting state, which indicates that the episode α occurs completely inside a certain window f, and hence, the frequency of α will be increased one. Subsequently, WINEPI assesses the frequency of the episode and determines its status as frequent or non-frequent by comparing it to the predetermined threshold *min_sup*.

To illustrate, Figure 3.4 provides a visual representation of an automaton designed to identify all occurrences of $\alpha = a \rightarrow c \rightarrow b$. In this context, ε comprises the event types set within the event sequence S. Regarding parallel episodes, as mentioned previously,

there are no constraints imposed on the relative order of events within the episode. The only one requirement is that all events constituting such an episode must occur within a given window to contribute to the increment of the support of the episode.

Several following algorithms have been influenced by the anti-monotonicity property of the episode's support (frequency), which is the foundation of WINEPI. However, the support definition based on windows introduces various challenges. A primary concern revolves around the issue of duplicate support counting, wherein, for a given episode α, the algorithm may consider multiple windows for a single occurrence. As showed in the sequence illustrated in Figure 3.1 with a window width $win = 4$, the serial episode $\alpha = a \rightarrow b$ supported 4 windows, even though it is evident that the occurrence of a followed by b transpires only twice. Consequently, the resultant frequency fails to accurately capture the events throughout the entire sequence. To rectify this concern, an alternative approach to WINEPI has been proposed, taking into account the minimal occurrences [12] of episodes. This alternative is designated as MINEPI (**MIN**imal occurrence-based **EPI**sode mining).

For further details on WINEPI, MINEPI, and other frequency definitions, interested readers can refer to [14, 13, 12, 50, 65, 44]. The pseudo code of WINEPI is provided in Algorithm 1.

Algorithm 1: The pseudo-code of WINEPI algorithm

Input: E - Event types set,
min_sup - Support threshold,
win - A window width,
S - Sequence of events over E.
Output: F - Frequent episodes set.

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

The WINEPI method is constrained by the inability to identify episodes comprising events

exceeding the window size. In response to this limitation, Casas-Garriga introduced a novel algorithm named EpiBF [76]. EpiBF is designed to unveil unbounded episodes by adaptively augmenting the length of the window contingent on the size of the episode, employing a parameter termed as **tus** (time-unit separation).

For every episode γ, the algorithm computes the support with respect to the window length, denoted as *win*, as determined by the formula: $win = (|\gamma| - 1) \times tus$. Considering a window width *win*, the support (frequency) of a given episode in a sequence S is expressed as follows:

$$fr(\gamma, S, maxwin) = \frac{|\{w \in W(S, maxwin) | \gamma \; occurs \; in \; w\}|}{|W(S, maxwin)|}$$

Here, the set of all windows of size *maxwin* is denoted by $W(S, maxwin)$.

3.3.2 FEM-DFS: A general depth-first search algorithm.

Depth-first search methods like FEMDFS [65], *Extractor* [82], MANEPI [55], WinMiner [62], and POLYFREQDMD [66] initiate their procedure by scanning the event sequence to identify all frequent 1-episodes. To gain insight into the functionality of these algorithms, it is beneficial to show the search space in the form of an episode tree, depicted in Fig. 3.3, wherein each node signifies an episode. The empty set is the tree's root, while episodes of size one are represented by the root's child nodes.

DFS algorithms are characterized by their utilization of a depth-first search strategy to navigate the tree. To enumerate episodes of greater size, Any DFS method, following lexicographical order [14, 13, 17, 83] (i.e., $A \prec B \prec C \prec \cdots \prec Z$), creates a novel episode or extension by joining a frequent episode (originally a 1-episode) with one of its siblings. This recursive process is iteratively employed to construct larger episodes, progressing down a specific branch of the tree, until further extension becomes impracticable. Following this, the algorithm backtracks to explore the generation of larger episodes from alternative nodes, employing the same iterative procedure.

The depiction of the depth-first search exploration process is provided in Fig. 3.3 (bold arrows). The initialization of the DFS search involves the computation of frequent episodes of size 1 set, denoted as $F = \{< a >, < b >, < c >, < d >\}$.The DFS approach then creates larger

episodes by extending each episode with 1-episodes (episodes of size 1). For example, $< a,b >$ is produced when episode $< a >$ and $< b >$ are combined. The method creates a new episode $< a,b,c >$ by merging the created episode with the single-event episode $< c >$, if the episode $< a,b >$ is frequent. If this episode proves to be frequent, it undergoes further extension with $< d >$, resulting in $< a,b,c,d >$. In instances where this episode is identified as infrequent, the algorithm retraces its steps along the same prefix and attempts to merge it with an alternate 1-episode. For instance, considering $\alpha =< a,b >$ and the and $\beta =< c >$, if the resultant $\gamma =< a,b,c >$ is identified as infrequent, the process preserves $\alpha =< a,b >$ and extends it with the $\beta =< d >$. This iterative process is systematically pursued to comprehensively explore the entire search space. Notice that each newly identified frequent episode is promptly incorporated into the result set.

Such a DFS approach generally keeps a significantly smaller number of episodes in memory at any given moment compared to breadth-first search algorithms, thereby reducing memory consumption. Additionally, the episode generation process can be more time-efficient.

While many algorithms yield satisfactory results under a specific support definition, it's crucial to acknowledge that altering the support definition may necessitate adaptations in the mining process. Consequently, these adjustments could lead to the generation of different number of interesting patterns. For example, a method that relies on *head frequency* produces a different set of frequent episodes if the support definition is switched to *window frequency* or any alternative frequency definition.

To offer a versatile approach accommodating various frequencies, Zhu et al. introduced a comprehensive approach named FEMDFS [65]. This algorithm is specifically designed for efficiently tracking the occurrences of frequent serial episodes, employing a definition of frequency based on episode's occurrences known as "earliest-transiting" occurrences[13]. Consider an occurrence h (as defined in paragraph 3.2.1) of a serial episode $\alpha = \alpha_1 \rightarrow \alpha_2 \cdots \rightarrow \alpha_k$. Let h be an occurrence of α, this occurrence is said to be an *earliest-transiting* occurrence of α if the initial occurrence time $t_{h(\alpha_i)}$ for each event α_i (where $2 \leq i \leq k$) follows the occurrence time $t_{h(\alpha_{i-1})}$ for the preceding event α_{i-1}. The *earliest-transiting* occurrences set of β is denoted by $eto(\beta)$. For instance, consider $\beta = c \rightarrow d$. The events $(c,25)(d,26)$ form an earliest-transiting occurrence of the episode β. The other two earliest-transiting occurrences of β are $(c,28),(d,30)$ and $(c,35),(d,36)$.

FEMDFS receives a sequence, a specified minimal support threshold, and a chosen frequency definition (referred to as *defsup*) as input parameters for the computation of episode support (e.g., total frequency, minimal occurrence-based frequency). The algorithm starts by extracting the earliest-transiting occurrences set, employing them to assess the episodes' frequency in accordance with the selected frequency definition. FEMDFS then engages in the search process by exploring the space using a depth-first strategy to uncover larger episodes. Following the concatenation of a pair of k-episodes, the set eto that represents the occurrences of the resultant (k+1)-episode is computed via FEMDFS. Next, it determines the subset of occurrences in accordance with the chosen frequency definition, denoted as *defsup*. A comparable approach is adopted by the FEM-BFS approach (as discussed in [65]), which employs a breadth-first search strategy and uses the earliest-transiting occurrences set to have a comprehensive overview of the event sequence for each ordered collection of events. The pseudocode of FEMDFS is depicted in Algorithm 2. Like other depth-first search algorithms, FEMDFS has demonstrated strong performance in experimental evaluations [65].

Algorithm 2: General Algorithm of FEM-DFS

Input: E - The event types
$minsup$ - Support threshold,
S - Event sequence,
$defsup$: Support definitions.
Output: F-set of all frequent episodes

```
2  Scan the sequence to obtain P the set of frequent episodes of size 1.
4  F ← P
6  for each frequent episode α ∈ F do
8      for each frequent 1-episode β ∈ P do
10         if def_sup ≠' to' then
11             // Grow the episode α with episode β
13             γ ← concat(α, β)
14         else
15             // Grow the episode β with episode α
17             γ ← concat(β, α)
18         // The earliest transiting occurrences of concat(α, β)
20         eto(γ) ← ComputeETO(eto(α), eto(β), defsup)
22         sup(γ) ← ComputeSup(eto(γ), defsup)
24         if sup(γ) ≥ minsup then
26             F ← F ∪ {γ}
28 return F
```

Frequent Episode Mining (FEM) algorithms vary in several aspects: (1) the adopted search strategy, either a depth-first or breadth-first search; (2) the approach to counting support for

determining episode frequency; specifically, the choice of a frequency definition; (3) the types of episodes considered for calculation; and (4) the method that helps in reducing the search space to optimize time and memory usage. Additionally, researchers have explored the design of FEM algorithms adapted for specific practical contexts, such as cloud environments [84, 22]. Furthermore, an important aspect is that many algorithms primarily focus on discovering sequential episodes, while real-world scenarios often involve numerous simultaneous events. Recent studies have initiated exploration into such cases, especially in applications like medical contexts [59]. This area of research needs to be further investigated. Additionally, most existing algorithms for episode mining have been developed for centralized systems. Thus, there is a growing challenge in designing algorithms suitable for distributed systems or creating parallel algorithms to enhance the flexibility and performance of the process of episode mining.

3.4 Extensions of the traditional FEM framework

While FEM finds diverse usages, its assumptions come with inherent limitations. This section initially examines the principal limitations of FEM and subsequently explores extensions designed to mitigate these constraints. These limitations are cited below:

- *An enormous quantity of generated patterns.* The size of frequent episodes found in the sequence is strongly influenced by the frequency threshold. With low support values, any FEM algorithm may explore a huge set of episodes, reaching into the millions. In such scenarios, algorithms not only exhibit prolonged runtime and high memory usage but also render the analysis of output challenging humans capabilities of understanding generated knowledge. Moreover, the collection of recurrent episodes can exhibit redundancy, as the frequency of an episode implies the frequency of its sub-episodes for several frequency definitions. Fortunately, there are algorithms dedicated to mining **"concise representations of episodes"** (refer to subsection 3.4.1). Reducing duplication within the collection of often occurring events while maintaining the vital information is the aim of these representations. Therefore, concise representations are considered as a condensed summary of the complete set of frequent episodes, and algorithms tailored for this purpose typically exhibit greater efficiency compared to traditional approaches.

- *Setting a threshold for minimim support difficulty* Another challenge with traditional FEM

algorithms is the difficulty in determining a suitable threshold value that yields an appropriate balance, neither too many episodes nor too few, but just enough. In the absence of background knowledge, users must resort to trial and error to identify an optimal threshold, and even a slight adjustment can lead to significantly different results. This fine-tuning process can be extremely time-consuming. In response to this challenge, an alternative approach involves mining the **top-k frequent episodes** (refer to section 3.4.2), where users specify the value of k, which take place of the frequency threshold.

- *The requirement for additional constraints*. It is often necessary to impose additional limitations on the frequency of episodes in time-sensitive settings. These limitations may have to do with things like the time interval between occurrence or the length of time occurrences themselves. A **constraint-based episode mining method** is an algorithm that can include these limitations (see sub-section 3.4.3).

- *Frequency is not always the most important criterion to select patterns.* In certain applications like market basket analysis, additional importance measures, such as utility, have been developed. Utility facilitates the evaluation of aspects like the profit generated by episodes rather than just their frequency. Sub-section 3.4.5 provides an overview of **high utility episode mining**.

- *Unsuitable to mine episodes in a dynamic environment.* In certain situations, the need arises to explore interesting episodes in a continuously updated event stream rather than a static sequence. Traditional FEM algorithms lack the capability to incrementally update episodes. Numerous algorithms have been introduced to address **the extraction of frequent episodes from dynamic sequences** (refer to sub-section 3.4.4).

- *All events are regarded as equally significant.* Traditional FEM assumes equal importance for all event types. But in real-world situations, this presumption is frequently untrue. Several techniques for **weighted episode mining** have been presented as a solution to this issue, allowing the assignment of weights to individual event types (refer to sub-section 3.4.6).

- *Lack of ability to deal with imperfect sequences.* Real-life event sequences frequently encounter imperfections, particularly in cases where event occurrences are **uncertain** (refer to sub-section 3.4.7) or **imprecise** (refer to sub-section 3.4.8).

3.4.1 Episodes with concise representations

Numerous studies have investigated the extraction of concise representations for episodes with the aim of minimizing the volume of identified episodes. These studies indicate that such representations can enhance accuracy across diverse prediction tasks [84, 22, 15, 82, 17, 85, 79, 62, 86, 87, 16, 52].

Table 3.2 presents the characteristics of key algorithms utilizing concise representations, focusing on the definition of frequency and the particular episode types that each algorithm targets.

Table 3.2: Summary of concise representations episode mining methods

Algorithm	Definition of frequency	Episode type	Concise representation
LA-FEMH+[22]	minimal occurrence-based	Serial	Maximal Episodes
MaxFEM[88]	head frequency	Serial and Parallel	Maximal Episodes
Extractor[82]	minimal and non-overlapped occurrence-based	Serial	Generator Episodes
FCEMinner[17]	minimal and non-overlapped occurrence-based	Serial	Closed Episodes
2PEM[15]	minimal and non-overlapped occurrence-based	Serial	Closed Episodes
WFECM	window-based frequency	Serial	Closed Episodes
MineEpisode[87]	minimal and non-overlapping occurrence-based	Serial and Parallel	Closed Episodes
CloEpi[16]	minimal occurrence-based	Serial	Closed Episodes
PPT/EPS[52]	minimal occurrence-based	Serial	Maximal Episodes

Widely employed representations for episodes comprise maximal episodes [22], closed episodes [84, 15, 17, 85, 86, 87, 16], and generator episodes. Certain representations, like generator episodes, have been utilized for extracting representative episode rules [82]. The subsequent definitions formalize each of these episode types. For clarity, the notation FE will denote the complete set of frequent episodes derived from an input event sequences.

3.4.1.1 Closed episodes

Closed episodes (CE) are often occurring episodes without an appropriate super-episode with the same number of supports, *i.e*:

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

By analyzing the forward, backward, or middle extensions of each common episode, algorithms like CloEpi determine the collection of often occurring closed episodes [16]. In simple terms, this is determining if an event may be added to such an episode before, within, or after it, implies creating a bigger episode with the same frequency. The episode is considered closed if this need is satisfied; if not, it is not. Investigating closed patterns instead of every frequent one can greatly reduce the output set while keeping all of the data. Without repeatedly scanning the sequence, all of the frequently occurring episodes may be reconstructed from the collection of closed episodes. Closed episodes can be used in situations where the definitions of frequency are anti-monotonic because of this characteristic. Newer algorithms designed for the extraction of frequent closed episodes comprise FCEMinner [17], CloEpi [16], MineEpisode [87], and 2PEM [15].

3.4.1.2 Maximal episodes

Maximal episodes (ME) [22, 88, 52] constitute the collection of episodes that lack any recurring super episode, expressed as:

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

An important feature of maximal episodes is that $ME \subseteq CE \subseteq FE$. As such, maximal episodes provide a more concise view than closed episodes. MaxFEM [88] and LA-FEMH+ [22] are two approaches designed in the aim of retrieving maximal episodes. These algorithms' main goal is to produce a more condensed collection of episodes. Given this background, an episode is considered *redundant* if it is a proper sub-episode of another frequent episode and is not included in the output. While the LA-FEMH+ algorithm focuses on discovering serial episodes, MaxFEM can identify both parallel and serial episodes.

3.4.1.3 Generator Episodes

Generator episodes (GE) are frequent episodes that exhibit unique support counts for their sub-episodes, and their super-episodes with equivalent support counts are closed [82]. Formally, this can be expressed as:

$$GE = \{\gamma | \gamma \in FE \wedge \forall \gamma' \sqsubset \gamma : \ sup(\gamma) \neq sup(\gamma') \wedge \exists \delta \in CE : \ \gamma \sqsubset \delta \ \wedge \ sup(\gamma) = sup(\delta)\}$$

Generator episodes do not offer a representation without loss of information about frequent episodes, in contrast to closed episodes. *Extractor* is a depth-first approach presented by Zhu et al. dedicated for prediction task in event streams [82]. This algorithm mines generator episodes using a depth-first search strategy, then derives a collection of representative episode rules. The process involves calculating the frequent episodes set based on a threshold for the frequency using a definition of frequency based on minimal and non-overlapping occurrences. Then, it verifies the closure of the retrieved episodes (refer to Sub-section 3.4.1.1). Afterward, the algorithm computes the generator episodes. Lastly, it creates a set of representative episode rules by using both the generator and closed episodes. This phase takes the sets of generator and closed episodes as inputs, together with the threshold value of confidence. In general, the task a representative episode rule is defined as follows: *"For a given episode rule γ, γ is considered representative if there is no other episode rule γ' with the same support count and confidence, where the antecedent of γ' is a subepisode of the antecedent of γ, and the consequent of γ' is a super-episode of the consequent of γ"*[82]. In order to provide meaningful predictions, these rules try to predict, given rules with the same support and confidence, the greatest amount of information (biggest consequents) from the smallest amount of information (smallest consequents).

Avinash et al. introduced another concept called *injective episodes*, defining an episode α as injective if it does not contain any repeated event types. Avinash et al.'s algorithm [51] is the exclusive method for discovering injective episodes. This algorithm mines injective episodes without order restrictions, employing a frequency based on non-overlapped occurrences. Furthermore, Avinash et al. introduced a novel episodes class called *chain episodes*, comprising both serial and parallel episodes, whether injective or non-injective. They also presented an effective approach for mining *chain episodes* [54]. While there are limited studies addressing advanced patterns in temporal data mining, like closed, maximal, or generator episodes,

this provides an great chance to move forward the state-of-the-art in episode mining as a field, particularly with concise representations.

To enhance the precision of the explored episodes, Tatti proposed an extra metric known as *episode significance* [89]. The calculation of an episode's significance serves as a post-processing step to evaluate the real interest of the identified frequent episodes. Significance computation is based on minimal windows, classifying a recurring episode as "significant" if its minimum windows' average durations are longer than what the independence model predicts.

3.4.2 Mining of the top-k frequent episodes

An additional aspect within the domain of episode mining is the mining of top-k episodes [90, 53, 91]. In this task, users have the ability to explicitly specify the number of episodes they need to identify. This extension arose from the recognition that determining an appropriate minimum threshold value can be a challenging task for users. When the threshold of frequency is established too low,the process of discovery yields an excessive number of frequent episodes, demanding substantial memory or leading to prolonged runtime. Contrarily, if the threshold is set too high, the process risks overlooking many episodes of significance. The iterative execution of the algorithm with varying threshold values to pinpoint the optimal number of episodes can be a time-intensive process.

The challenge posed by determining an appropriate number of episodes is addressed by top-k episode mining algorithms, allowing users to directly specify a parameter k, representing the desired episodes number to be identified. Consequently, the top k most recurring episodes are provided to the user via a top-k algorithm. This approach offers the advantage that users are spared the need to run the algorithm multiple times while adjusting the minimal limit to achieve a specific episodes number.

However, the main issue of episode discovery using top-k approaches arises when attempting to generate the same number of episodes as traditional FEM with equivalent settings (k and the minimum threshold) [90]. The complexity comes from the fact that, in top-k episode mining, in order reduce the search space, there is no presumption regarding the support of the top-k episodes. Consequently, top-k algorithms usually begin the episode search with an internal minimal value of threshold for the frequency of 0. Upon finding k episodes, the approaches

increment the internal threshold and carry on searching. The top-k episodes are given output to the user when no further episodes are found. Top-k episode mining has to search a wider area in order to find the same amount of episodes as a regular FEM algorithm because of this search procedure [90].

3.4.3 Episode mining with constraints

Numerous techniques have been developed for mining episode with more constraints, a process that incorporates more than one constraints into Frequent Episode Mining (FEM) to eliminate uninteresting episodes. Constraints, specific criteria set by the user, define the conditions that frequent episodes must meet. Numerous criteria have been put out in the research field, and they are often used to either find frequent episodes or as a post-processing step on the output of a FEM method (removing episodes that don't fit the criteria).

For maximum efficiency, it is preferable to incorporate constraints into the search process for episode mining. This strategy has the potential to diminish the search space, rendering constraint-based algorithms potentially faster and more memory-efficient in comparison to traditional Frequent Episode Mining (FEM) algorithms. The extent of efficiency gains is contingent on the specific constraints selected.

One prominent algorithm in the field of constraint-based episode mining is *DiscoveryTotal* [14]. This algorithm introduces two types of constraints. The first constraint, termed the *span constraint* or *expiry-time constraint*, takes into account the maximum allowable time between the first and last events of an episode's occurrence. By introducing the span constraint, the algorithm diminishes the frequency of episodes by eliminating occurrences with events spaced too far apart in time. The second constraint pertains to the maximum allowable time between two consecutive events, referred to as the gap or inter-event constraint. Several algorithms, including WinMiner [62], EPS, and PPT [52], have taken into account such time constraints. Furthermore, scholars have investigated constraints associated with the minimum and maximum number of events per episode, known as the length constraint, in the context of analyzing heterochrony in developmental biology [92]..

3.4.4 Mining episodes from dynamic sequences

Traditional episode mining techniques face a limitation in handling dynamic sequences as they assume a static input. These algorithms are typically designed to process a single, lengthy event sequence in one go, yielding relevant episodes. However, in the case of newly logged events or revisions to information about previous events, such a traditional algorithm must be rerun entirely to capture the latest frequent episodes. In such scenarios, traditional frequent episode mining (FEM) tasks prove inefficient. Fortunately, several algorithms have been developed for online and stream episode mining [93, 82, 94, 59, 79, 95, 40, 96, 97, 98, 99, 100]. Zhu et al. developed the *Extractor* algorithm with a focus on mining episodes within event streams and generating prediction rules [82]. The algorithm includes identifying closed episodes and their generators within a log of events and subsequently creating non-redundant rules (representative rules) applicable to the stream of events. To overcome the difficulty of extracting episodes from event streams, Xing et al. developed the *MESELO* approach ("**M**ining fr**E**quent **S**erial **E**pisodes via **L**ast **O**ccurrence") [97]. This algorithm identifies episodes within a stream of events by grouping events within small groups, capturing and storing only the latest incoming batch at any time. The input parameters for MESELO include an event-growing sequence S, a support threshold $minsup$, a maximum window size δ, and a window size δ for the current valid sequence. The purpose of "Online Frequent Episode Mining" (OFEM) is to extract all frequent episodes from S with the latest timestamps within δ, ensuring that the size of each episode's occurrence is no greater than δ, and that the frequency of the episodes is at least $minsup$. Additionally, approaches like ONCE and ONCE+ [79] have been developed specifically for mining serial episodes with temporal constraints in streaming data by focusing on the last occurrence of each episode.

Besides, Xing et al.[97] defined the problem of online episode mining and proposed the MESELO algorithm (**M**ining fr**E**quent **S**erial **E**pisodes via **L**ast **O**ccurrence). The input for this algorithm includes an event-growing sequence S, a support threshold $minsup$, and maximum window sizes δ and Δ for the current valid sequence. The novel challenge in Online Frequent Episode Mining (OFEM) involves extracting all frequent episodes from S, ensuring that it contains the latest timestamps within Δ. Additionally, the size of each episode's occurrence should not exceed δ, and the frequency of the episodes should be at least $minsup$.

Various extensions of episode mining delve into more intricate event sequence represen-

tations, going beyond of traditional approaches. In the subsequent sections, we present an overview of the most prevalent extensions in this domain.

3.4.5 Mining high utility episodes

One of the limitations of traditional FEM is that it only takes into account the number of occurrences, not their relative importance. In reality, not all event types hold the same level of significance. High utility episode mining (HUEM) extends traditional FEM by incorporating the utility or importance of events to identify significant episodes [33]. Algorithms designed for High Utility Episode Mining take a complex event sequence as input, allowing events to occur simultaneously. Additionally, information about both events' external and internal utility is considered. "External utility" signifies an event type's proportional relevance throughout the sequence, while internal utility indicates the event's importance at the time of occurrence. Every utility value is represented by a positive integer. The objective is to identify episodes with utility that exceeds a specified minimum value, referred to as "high utility episodes".

High-utility episode mining (HUEM) presents a challenge due to the non-anti-monotonic nature of utility calculation functions, making them unsuitable for direct use in search space reduction. The pioneering algorithm, UP-Span [33], was introduced to discover all high utility episodes along with their minimal occurrences. It employs an anti-monotonic upper bound on utility to mitigate the search space. Some algorithms, namely TSpan [34] and HUE-Span [35], have been proposed as improved solutions to further reduce the search space and enhance performance, particularly when dealing with extensive databases.

An application of High Utility Episode Mining (HUEM) explored in previous studies is its use in stock investment prediction. A methodology was introduced that combined HUEM with a genetic algorithm, yielding superior results compared to a method based on frequent episodes [101].

3.4.6 Mining weighted episodes

Mining episodes based on weights is an extension of FEM, focusing on extracting episodes from multiple sequences. The introduction of *weight* in this extension serves the purpose of quantifying the relative significance of each sequence. The framework of weighted episode

mining can be considered as a variant of utility-based episode mining, with the distinction that, in the utility model, each event is assigned a weight. The only existing study of weighted episode mining was proposed by *Liao et al.* [85]. The problem is defined as follows: Given D a collection of sequences, a maximal window width *maxwin* specified by the user, *minsup* and a parameter $minsup\_wa$. The objective is to discover all frequent closed episodes with weights in D, where each episode $P =< e_1, e_2, \ldots, e_k >$ must fulfill the following conditions: :

i) t_k - $t_1 \leq max\_win$,

ii) There is at least a sequence $S_i \in D$, such that $sup_i(P) \geq min - sup$, 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 represents the number of sequences that contain the episode P.
3. N is the total number of sequences.
4. Ω_i represents the weight of the i^{th} event sequence of D.

For example, consider the set of three sequences (S_1, S_2, and S_3 with weights 1, 2, and 3, respectively) depicted in Fig. 3.5 (N=3). Consider the episode $P =< CDE >$. It occurs in S_1, S_2, and S_3 a respective 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, adhering to the support definition outlined in [85]. Thus, its weighted support is computed as $sup\_wa(P) = \frac{3}{3} \times (1 \times 3 + 2 \times 3 + 3 \times 4) = 21$. Consequently, P is a weighted frequent closed episode, as $sup\_wa(P) \geq minsup\_wa$.

Generally, weights in weighted episode mining can be assigned through diverse methods, contingent on the application, and exploring the design of such methods presents an avenue for further investigation. Potential approaches may involve manual assignment based on statistics, external data, or past experience.

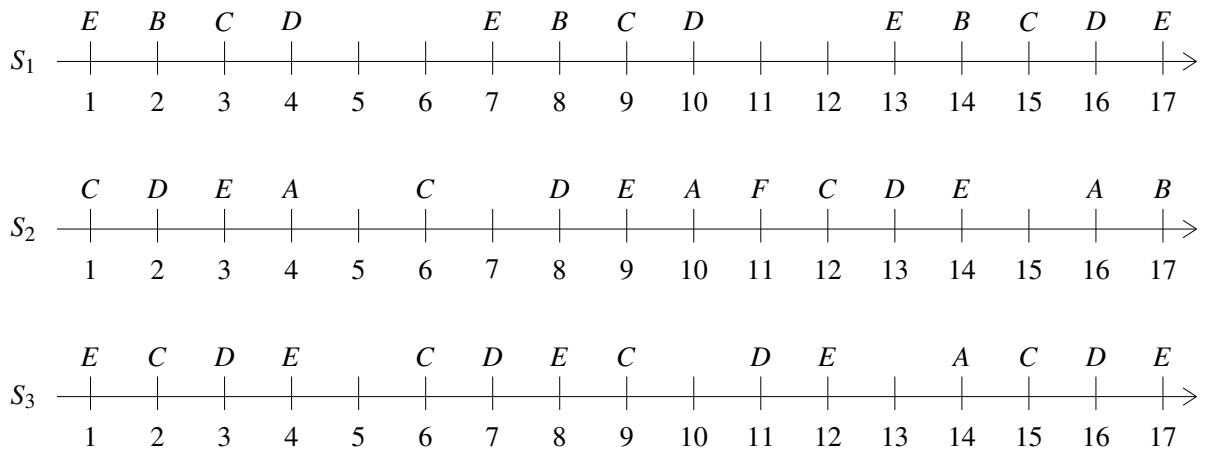


Figure 3.5: A collection of event sequences

Assume that P only appears in S_1 10 times ($N_I = 1$). Hence, the weighted support will be $sup\_wa(P) = \frac{1}{3} \times (1 \times 10) = 3.3$. As a result, episode P would not qualify as a weighted frequent closed episode.

3.4.7 Uncertain episode mining

Mining uncertain episodes extends the scope of episode mining, designed to handle uncertain data. In various applications, events collected from sensors may be noisy, or the events themselves may be inaccurate, rendering traditional episode mining less effective for analysis. This extension introduces a probabilistic episode mining model as the main uncertainty model. In this model, users define two thresholds: a minimum probability threshold, indicating confidence in the frequency of episodes, and a support threshold to verify if generated episodes are likely frequent. Specifically, A specific episode's frequentness probability needs to be at least as high as the probability threshold.

In the discovery of uncertain episode context, events undergo a redefinition, and each event is assigned a probability value denoted as p within the interval $[0,1]$, representing the likelihood of the event's presence in the sequence. The resulting sequence is called *"uncertain sequence"*, where events are called *"probabilistic events"*, and Li et al. have formally defined these concepts. They introduced methods for identifying probabilistic frequent serial episodes (P-FSE) [73], comprising: (1) an "**exact approach**" determining the precise support value of a specified episode, (2) an "**approximate approach**" estimating the support through a distribution of probabilities, and (3) an "**optimized approach**" offering a maximal limit on the support with

no accounting for every occurrence of an episode.

P-DFSE [72] is another algorithm tailored for uncertain episode mining, specifically designed to discover probabilistic frequent serial episodes by taking into account the relationship among occurrences of a specific episode. Both algorithms employ the "possible world semantics" theory, treating each occurrence of an episode as a possible world. An episode is considered as a "probabilistic frequent episode" only when the probability computed in a possible world, as described in both [72] and [73], exceeds the specified probability threshold.

3.4.8 Fuzzy episode mining

It is a task that extends traditional episode mining by incorporating fuzzy sets theory to handle imprecise events. In this task, event sequences are denoted as $S = \{E_1, E_2, \ldots, E_n\}$ with associated attributes $A = \{a_1, a_2, \ldots, a_m\}$. The representation of every event E is $E = \{E.a_1, E.a_2, \ldots, E.a_m\}$, where m values are contained and an integer T indicates the event's position in the sequence S. There is only one algorithm available for mining fuzzy episodes was introduced by Luo et al. [57]. In this research, each episode is defined as $P(e_1, e_2, \ldots, e_k)$ as a set of event variables, with $e^q\{attr_1 = v_1, \ldots, attr_q = v_q\}$ representing an event variable with q attributes. Since attributes can be quantitative (fuzzy) or categorical, each attribute value has its membership degree based on a fuzzy set or category. The product of the event variables in any event determines the occurrence of the episode E. Luo et al.'s algorithm utilizes the episode mining algorithm of Mannila et Toivonen, which focuses on minimal occurrences [50]. The task of Luo's algorithm is: given a sequence of events and a minimum occurrence threshold *min_occ*, fuzzy episode mining aims to identify all episodes with an occurrence frequency exceeds *min_occ*. This study demonstrated the effectiveness and the practicality of fuzzy episode rules in instant identification of intrusions.

3.5 Further problems related to episode mining

We have discussed the episode mining problem and a few of its extensions in the previously stated subsections. This subsection explains some more pattern mining challenges that have a close relationship to the episode mining problem.

Episode Rule Mining or (ERM). It is a very interesting extension of episode mining. An

implication $\gamma \rightarrow \delta$ is known as a **episode rule**. where γ and δ are two common episodes. This means that upon observing an episode γ, it is probable that δ will follow. The precise definitions of γ, δ, and the relationship between them can vary among algorithms, influencing their interpretation and practical applications. The discovery of episode rules is valuable for predicting future events, understanding the data, and making informed decisions.

The concept of episode rules shares similarities with association rules in traditional pattern mining [2], where the goal is to discover association rules itemsets in a table that is binary. Nevertheless, one important difference between episode rules and association rules is that the former are formed from temporal data, namely an event sequence, whereas the latter do not take time into account. Consequently, episode rules are useful in identifying relationships across time.

The task of obtaining all episode rules is referred to as *episode rule mining*. Mannila et al. [12] offered the WINEPI and MINEPI algorithms in their work, which proposed the first approach for mining episode rules. It identifies recurring episodes and then pairs them to create episode rules. The **confidence** of a rule is a measurement that describes its interestingness, it is computed and evaluated with respect to a threshold for the confidence. If the confidence of the generated episode rule is not less than a threshold min_conf, The user can use the given rule to forecast future occurrences because it is regarded as valid. Numerous episode rule mining have been proposed[102, 58, 82, 103, 104, 62, 49, 63, 64].

For episode rule discovery, the first method, MINEPI (discussed before), uses a breadth-first search strategy. WinMiner [62], on the other hand, efficiently mines rules under a window constraint by using a depth-first search strategy. Additionally, many algorithms, like DEER [103], perform episode rule discovery with no need for generating candidates. Several algorithms have been developed to evaluate event streams [102, 82] and uncovering episode rules from event sequences based on its utility [64]. The extraction of episode rules finds applications in diverse areas, including telecommunication alarm analysis [12], detection of intrusions [46], and the detection of internet anomalies [49].

However, a challenge in episode rule mining arises from the need for a strict ordering between events. Consequently, an algorithm may generate numerous rules that are subtly different but essentially describe the same real-world situation. To address this issue, another

type of episodes, known as partially-ordered episode rules, is employed to capture relationships between events. A recent algorithm, POERM [67], has been introduced for mining partially-ordered episode rules to unveil relationships in complex sequence of events. POERM efficiently identifies rules of the form $A \rightarrow B$, where A and B are event sets. The problem addressed by POERM is as follows: Given a complex event sequence S, three integer values $ASpan$, $BSpan$, and $ABSpan \in \mathbb{Z}^+$, along with minimum thresholds for support and confidence $minsup$ and $minconf$, a rule $A \rightarrow B$ is considered valid if: (1) every occurrence of A and B has a maximum duration that doesn't go beyond $ASpan$ and $BSpan$, respectively; (2) the time interval that exists between any pair of occurrences of A and B do not exceed $ABSpan$; and (3) $sup(A) \geq minsup$ and $sup(B) \geq minsup \times minconf$.

Several studies have explored alternative approaches to calculate the episode rule's confidence value. One notable contribution is the MARBLES framework [105]. The study introduces a set of episode rule mining algorithms. MARBLES aims to extract all rules by using three new frequencies by employing *fixed windows*, *minimal windows*, and *weighted minimal windows-based* frequencies. Furthermore, in order to find compelling episode rules, MARBLES adds *minimal-extensibility* and *weighted-extensibility* for minimal window-based frequency and weighted frequency, respectively. This framework introduces a new definition of confidence adapted for such frequencies, aimed at capturing episode rules that maximize confidence.

While many existing studies focus on identifying interesting episodes in simple or complex sequences, the primary objective of discovering interesting knowledge is to leverage this knowledge for detecting patterns that depend on one another, helping in understanding the past of predicting the future actions. Consequently, episode rules are the best patterns that encapsulate these relationships. Research in this field is ongoing with the goal of creating new algorithms for the extraction of these rules in various settings and their interpretation for use in real-world scenarios.

Periodic episode mining. It is an alternative to episode mining [40]. The objective is to identify episodes that exhibit both frequent occurrence and periodicity in an event sequence or stream. To detect periodic patterns, the period length of each potential episode is computed as the time interval between successive episode's occurrences. If the period of an episode is less than a certain threshold, it is deemed periodic.

3.6 Recent applications of episode mining in various domains

As per previous subsections, it is evident that FEM framework is an effective method for analyzing sequential data. It is widely used and has been employed in recent studies to improve the functionality of several frameworks. Despite this, most recent research focuses on developing new algorithms based only on theoretical considerations, and the creation of interactive episode mining tools is still an active area that needs more investigation. Fortunately, the interactive tool SNIPER [106] is available for analyzing and understanding the performance of a recommendation system through episode mining. This tool empowers users to engage with the system and provide feedback on the quality of recommendations. Multiple systems have been created using episode mining principles along with corresponding interactive tools.

Frequent episode mining approaches have extended their applications to novel domains, including cyber-physical systems in intelligent cars [107] and In-Situ Decommissioning sensor network monitoring [108, 109]. They use a clear model of mechanisms of predicting and exploring frequent episodes, respectively. Every system incorporates a dedicated frequent episode mining (FEM) algorithm, explicitly applied to examine all gathered data and express the sequential links actions one to another in every platform. Nevertheless, the sequence creation step of the framework is where the two frameworks diverge. In the subsequent subsections, We outline the processes of these two common episode mining applications and provide examples of how episode mining was used to accomplish the stated goals.

3.6.1 Predicting events in cyber-physical networks

The goal of the Cyber-Physical System (CPS) is to enable intelligent user interactions by tightly integrating cyberspace and physical components. Examples of common cyber-physical systems (CPS) are autonomous vehicles, smart cities, and intelligent houses.A vast amount of data from these frameworks may be used for developing reliable tools, either to improve system performance or for predictive purposes.

Hence, an effective framework was introduced for predicting the future action in a Cyber-Physical System (CPS), such as vehicle use, as demonstrated in [107]. The approach consists of analyzing sequential events to predict future events through the utilization of a graph data structure that facilitates the construction of event chains (episodes). The framework consists

of three primary layers, with the fourth layer comprising the implementation of the model of prediction.It is feasible to integrate applications like intelligent control, system monitoring, and decision aids at the fourth layer. The three stated levels are explained in more detail below:

- **Extracting event instances.** It is the framework's first layer. It involves extracting instances of events (for simplicity, we'll called them *events*) from incoming data that come from various physical components. Every device links each event with its proper occurrence time. An event is constructed of a number of features, and *Linked Data* is used to represent it. This is formed in the following manner: $event =< URI, source, Attr >$ where $Attr$ is a tuple (A, S, O, D, T, P) (A:action, S:subject, O: Object, D:Device, T:Time, P: place), $source$ is a link to a collection of data from which the event is retrieved, and URI is a unique number that identifies the event.

- **Extracting event graph.** The relationships between events can be derived sequentially after extracting the event instances themselves. In the preceding layer, we demonstrated the association of each event instance with its corresponding occurrence time. The start and end times are included in the time field. Consequently, all events can be arranged into a single long sequence of events, enabling the application of a Frequent Episode Mining (FEM) framework to retrieve various relationships between the events within the built sequence.

  Therefore, a sequence ES from a set of events $IS = [e_{i_1}, \ldots, e_{i_n}]$ is expressed as follows: $ES = \langle (e_1, e_1.str), \ldots, (e_n, e_n.str) \rangle$, where e_i denotes the i^{th} event in the sequence, and $e_i.str$ denotes the starting time of e_i. After that, the sequence ES is analyzed to identify interesting episodes using an effective FEM technique called MANEPI [55]. In short, MANEPI is an efficient algorithm for mining frequent episodes within a long sequence, using a depth-first strategy method for calculating the frequencies based on minimal and non-overlapping occurrences. In addition, the discussed application seeks to apply MANEPI for mining frequent episodes while adhering to a time gap constraint, with the intention of reducing redundancy in the set of frequent episodes. This method is categorized as a constraint-based algorithm for episode mining since it uses a gap constraint. At last, this system constructs the event graphs for every serial episode using sequential relations and the associated transition probability among any consecutive actions (events),

given by:

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

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

- **Event prediction on the event graph:** In this stage, a prediction model is constructed using the event stream and the event graphs that were recovered from the preceding layer. To accomplish this objective, at first, candidate events and event contexts are recognized by the system. The events that have happened are referred to as a *event context*. An event context that could replace this one is represented by a *candidate event*. In this layer, predicting the events that will follow is the focus of the second step. Therefore, a three-stage structured model called SGNN is built in order to complete this step.

The three previously mentioned layers form the integral components of the framework. Another layer constitutes the potential application. Consequently, the application of the framework is applied to car usage, serving as an application layer. Additionally, it can integrate into Internet of Things (IoT) systems, including applications in smart cities. The prediction mechanism is useful in many areas, including early warning systems, intelligent control, behavior monitoring, and decision support.

3.6.2 In-Situ decommissioning sensor network monitoring using FEM

A collection of sensors installed at a nuclear facility known as In-Situ Decommissioning (ISD) is the subject of a recent application using episode extracting as an analytical tool. Techniques for mining episodes are used to examine the vast amount of data produced by In-Situ Decommissioning or (ISD) sensors [108, 109] since the collected data are time-specific, includes a timestamp attached to every event to indicate when it happened. ISD sensors are used to monitor the nuclear facility.

As previously stated, the information gathered from the sensors at the nuclear site may be thought of as a unified, continuous stream. These information comprises large records of (1)

details about the voltage of the battery, (2) localized strain gauge pressure data, (3) 4 thermocouples' worth of temperature data, and (4) information obtained from the tiltmeter (including tilt degrees and local temperature). There are two event types for each sensor in the network: **High** (H) or **Low** (L). For instance, taking into account a thermocouple among these four ones (t_1), where the recorded temperature exceeds the adjacent data point, it is categorized as an event of type t_1H (high temperature). If the recorded value is less than the data point, it is labeled as an event of type t_1L (low temperature). As a result, the events from the four thermocouples form the set $[t_1H, t_1L, t_2H, t_2L, t_3H, t_3L, t_4H, t_4L]$. Likewise, The remaining sensors are treated with the same idea, like for strain data $[SH, SL]$, for biaxial tiltmeter$[BH, BL]$, and for the information of the battery (voltage) $[VH, VL]$. The sensors record the time at which every event occurs in addition to the timestamps.

Moreover, the proposed method makes use of the FEM algorithm described in [83]. This approach uses the frequency definition based on non-overlapping occurrences. In order to shorten the time taken by the analysis process to look for frequent episodes and improve the mining program's performance, non-overlapping occurrences are specifically adapted for temporal analysis in the ISD sensor network. The resultant outcome of this application showcases of the FEM platform in timed data analysis effectiveness.

3.6.3 Intelligent coordination platform for wheel manufacturing

In recent decades, the Internet of Things (IoT) has found growing applications in numerous complex systems, comprising a vast network of interconnected devices that collect extensive data. Various complex systems collaborate to execute a common task, increase a process's effectiveness, or create novel features by collaborating with existing services from multiple systems. Implementing services within IoT settings presents a range of obstacles that must be resolved in order to achieve the desired outcomes. Three significant challenges in service collaboration are heterogeneity, variable invocation logic, and spatiotemporal restrictions. Zhu et al. presented a thorough architecture for collaborative services in the wheel manufacturing process in order to overcome these issues [110].

The presented framework shares similarities with the one proposed for Cyber-Physical Systems in [107]. Typically, the framework consists of the subsequent tasks:

- **Event instance extraction** from multi-source IoT data. During this stage, the Event Logic Graph (ELG) is represented by the framework as linked data, with each ELG having the following characteristics: $ELG \leftarrow (relations, events, url)$. Here, the event set and its relationships are indicated by the variables *events* and *relations*, respectively, while *url* serves as a unique identifier to identify the ELG. This framework proceeds to **create event instances** through a dispatcher, which supplies detectors capable of recognizing such events. For instance, the dispatcher might provide a detector specialized in detecting the activation of a particular machine and another detector designed to identify temperature increases. Notice that the detector could be a machine learning model or a rule engine.

- During the second phase, the platform organizes all event instances into a sequence based on their chronological order, constructing the event logic graph. After that, an effective framework for episode mining is applied to retrieve all significant episodes from the consecutive data. The framework uses an approach based on the well-known MANEPI algorithm [55] to achieve this goal. This algorithm has also been applied in the field of Cyber-Physical Systems, as was previously mentioned [107]. The novel method imposes both a time gap and a duration criteria to identify occurrences with a finite time interval between them and events that are not too long. Consequently, event chains collection (serial episodes) is devoid of redundancy.

- The last two tasks pertain to the practical application aspect, focusing on predicting future events. These tasks aim to use the prediction model in several real-world contexts, such as dynamic service composition, service suggestion (recommendation), and orchestration optimization.

The three application fields above highlight the effectiveness of Frequent Episode Mining (FEM) as a tool for temporal data analysis. Current approaches may find use in a number of different fields. As an illustration, NonEpi [43] proved to be an interesting algorithm for predicting diseases based on the observed symptoms. In order to accomplish this, the episode rules of the form $\gamma \rightarrow \delta$ (where γ is the predecessor of δ)must be unearthed. The advantage of extracting patterns in the form of rules is their human interpretability.

3.6.4 Mining train delays

A country's essential infrastructure includes a railway network, and any delays can lead to significant disruptions in railway transportation, including shutdowns. To develop an intelligent solution for predicting train delays, the CloEpi episode mining algorithm was employed to uncover hidden patterns in sequential train data obtained from Infrabel [111].

Before applying episode mining, the data undergoes preprocessing to generate a single, continuous sequence of events. The output of this step consists of a series of event sequences that specify the delays encountered by the train at each characteristic point. Every event in these sequences is represented by *(Train_id, timestamp)*, including a unique integer value *Train_id* that serves as representation of the delayed train, and *timestamp* which signifies the specific point in time at which the train experienced the delay. Finally, in order the mines train delays episodes, the well known CloEpi algorithm [75] is applied to analyze these sequences, which discovers closed episodes to minimize the size of the output without sacrificing any information.

3.7 Conclusion

We have presented in this chapter an overview of the frequent episode mining framework and its main concepts. Then we presented a classification of existing algorithms into different categories according the search strategy and the frequency definition. We have also presented the limitations of the traditional episode mining algorithms face to some problems to overcome. Then we have presented the different extensions presented for traditional algorithms in order to deal with these limitations.

In subsequent chapters, we will elucidate our contributions that have been advanced to address research prospects and remedy some deficiencies within this domain.

Chapter 4

Episode rules discovery under non-overlapped occurrences of frequent episodes

The previous chapter presented the state-of-the-art in terms of frequent episode mining (FEM) framework and its extensions, which involves various algorithms aiming at extracting useful knowledge from sequences. We also highlighted the various definitions of frequent episodes that depend on the occurrence of these sequences. The state-of-the-art of FEM includes numerous extensions that enable the extraction of episode rules, but the interpretation of these rules can be challenging when the occurrences overlap. Therefore, we proposed a new contribution called NonEpi, which is based on non-overlapping occurrences and mines frequent episodes and episode rules from simple event sequences. This chapter explains the new contribution by detailing the relevant concepts, taking into account those previously introduced such as **event**, **event sequence** and **episode** (as discussed in Section 3.2.1).

4.1 Introduction

FEM methods have been developed to derive interested episodes using several frequency definitions to track their occurrences. Mainly, we can categorize them into dependent or independent frequency. When episode's events overlap (the occurrences have some shared events) like *total frequency*, *head frequency minimal occurrence-based frequency*, the definitions are

categorized as **dependent frequency definitions**. The concept of the **independent frequency definition** prohibits overlapping occurrences, in contrast to the frequency definition based on *non-overlapped occurrences* and *distinct occurrences* (see table 3.1).

Previous research has focused on identifying strong episode rules in complex event sequences using various algorithms. Frequent Episode Mining (FEM) has been expanded to extract episode rules that satisfy a minimum confidence criterion. Other approaches have been developed to determine episode rules by first identifying interesting episodes using a proper frequency definition. In this chapter, we will explain the design of an efficient approach that mines episode rules based on a definition of frequency which tracks all non-overlapping occurrences. We will give thorough explanations of both the method and the underlying ideas in the sections that follow.

4.2 Basic definitions

The following subsection provides definitions of terms related to episode rule mining and episode mining based on non-overlapping events. As covered in the chapter before, the purpose of this approach is to extract injective serial episodes. As a result, in the parts that follow, a serial episode made up of injective nodes is just called an episode. An episode consists of three events ($\alpha = A \rightarrow N \rightarrow B$) that occur in the sequence shown in Fig. 4.1. Event A is followed by event N and then by B.

The above stated concept of a sub-episode applies to any type of FEM algorithm. But, in order to fulfill the features of a given algorithm, the mapping function between the episode and its sub-episodes is defined. As a result, we use the next definition of sub-episodes:

Definition 1 (Sub-episode)**.** *Consider two episodes $\alpha = \alpha_1 \rightarrow \cdots \rightarrow \alpha_n$ and $\beta = \beta_1 \rightarrow \cdots \rightarrow \beta_m$. The episode β is considered as a **sub-episode** of α (expressed as $\beta \sqsubseteq \alpha$) if and only if there exists an index k ($1 \leq k \leq n-m+1$) such that for all j ($1 \leq j \leq m$), $\beta_j = \alpha_{k+j-1}$.*

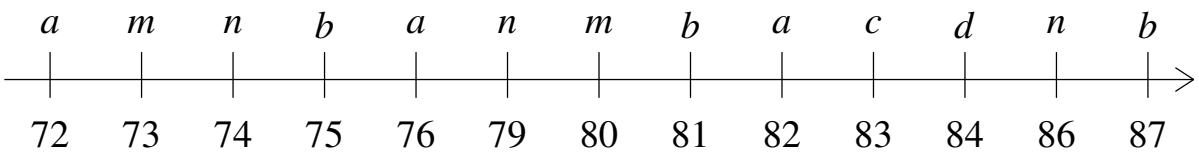


Figure 4.1: A sequence of 13 events

If β is a sub-episode of α then, each occurrence of α contains an occurrence of β [12]. We further refer to α as a **super-episode** of β.

There are two distinct situations identified: β is called a **prefix** of α when $k = 1$, i.e., when it is positioned at the beginning of α. In contrast, β is placed at the end of α and is referred to as a **suffix** of α if $k = n - m + 1$.

The episode $\beta = a \rightarrow n$ in the running example (see Fig. 4.1) is a sub-episode of the episode $\alpha = a \rightarrow n \rightarrow b$, which may be represented as $\beta \sqsubseteq \alpha$. In addition, β here is a *prefix* of α.

Definition 2 (Occurrence of an episode)**.** *In the sequence S, an occurrence of an episode α is represented by the vector $h = [t_1 t_2 ... t_n]]$. such that each t_i is an integer representing a time at which the i^{th} node of episode γ occurred (timestamp). Consider two occurrences $h = [t_1 t_2 ... t_n]$ and $h' = [t'_1\ t'_2\ \ldots t'_n]$, if $t_n < t'_1$ or $t'_n < t_1$, then h and h' are regarded as non-overlapped occurrences of episode α.*

For a particular episode, several sets of non-overlapping occurrences may generally be determined. Maximal sets, or sets with the greatest number of non-overlapping occurrences, are of special importance to us:

Definition 3 (Maximal set of non-overlapped occurrences of an episode)**.** *Let γ be an episode and H be its non-overlapping occurrences set, H is considered as non-overlapping occurrences maximal set if, for every other set H' of non-overlapping occurrences of γ, the condition $|H| \geq |H'|$ holds. As a result, the maximal set of non-overlapping occurrences of the episode α is denoted by $no(\gamma)$.*

Consider the sequence S shown in Fig. 4.1 as an example, as well as the episode $\gamma = a \rightarrow n \rightarrow b$. Because $no(\gamma) = H$, the maximal set of non-overlapping occurrences of α in S is set H = {[72 74 75],[76 79 81],[82 86 87]}.

Definition 4 (Support of an episode)**.** *The support of an episode γ (denoted as $support(\gamma)$) is the size of the maximal set of non-overlapping occurrences, expressed as $support(\gamma) = |no_\gamma|$.*

An episode γ is considered frequent based on non-overlapping occurrences if $support(\gamma) \geq minsup$, where $minsup$ represents a user-defined support threshold. The support of the episode $\gamma = a \rightarrow n \rightarrow b$ is 3, as we can see in the previous case.

Episode rules, as previously stated, capture binary associations between pairs of often occurring episodes. Recall that an expression of the type $\beta \rightarrow \alpha$, in which both α and β reflect frequent episodes, constitutes an episode rule. The so-called *confidence* of a rule is defined by its conditional probability. It is expressed as follows: $conf(\beta \Rightarrow \alpha) = \frac{support(\beta \sqcup \alpha)}{support(\beta)}$. The types of rules and how often they are used determine the exact meaning of the $\sqcup$ operator. In our study, we have $\beta \sqcup \alpha = \alpha$ since β is a sub-episode of α (see Section 4.3). If the rule's confidence is not less than the user-specified minimum confidence threshold, $minconf$, it is deemed legitimate.

Depending on the application area, many types of rules can be constructed, each encapsulating unique relationships between an antecedent and consequent. This chapter focuses on a particular class of episode rules that are useful in making predictions.

4.3 Definition of the problem

To calculate a rule's frequency, the NONEPI technique uses a frequency definition based on non-overlapping occurrence. In this context, an episode rule is defined as follows:

Definition 5 (Episode rule)**.** *An episode rule takes the form $\gamma \Rightarrow \delta$, with both δ and γ representing frequent episodes under the non-overlapping frequency condition, and γ being a prefix of δ.*

An episode rule is significant because it implies that the presence of the prefix β in the sequence has a significant impact on the occurrence of every subsequent event that forms an instance of α. This knowledge can be useful for predicting the future course of a series of ordered occurrences or identifying the fundamental cause of a specific phenomenon.

Therefore, the ratio between the support of δ and the support of γ is used to determine the confidence of an episode rule $\gamma \Rightarrow \delta$:

$$conf(\gamma \Rightarrow \delta) = \frac{support(\delta)}{support(\gamma)} \tag{4.1}$$

Finding all valid episode rules quickly and with the least amount of memory and time consumption is the main goal of an episode rule mining method. Consequently, the following is

how this algorithm defines the problem it addresses: "*The objective is to extract a set of valid episode rules (rules in the form of* $\beta \Rightarrow \alpha$ *such that* $conf(\beta \Rightarrow \alpha) \geq minconf$) given an event sequence S, support threshold $minsup$, and confidence threshold $minconf$."

4.4 The proposed approach for mining episode rules

We in troduce in this section the NONEPI approach, which uses non-overlapping occurrences to extract the collection of frequent episodes and episode rules. The overal approach is is divided into two stages: The first stage aims at finding frequent episodes and the second stage focuses on finding the set of valid episode rules.

4.4.1 Extracting frequent episodes

The first stage is to identify frequent episodes based on their non-overlapping occurrences as defined by Wan et al. in [72], where the input is the minimum frequency threshold $minsup$, which serves as a blueprint for this function. The production of frequent episodes under non-overlapping occurrence-based frequency depends on the following anti-monotony characteristic, which avoids the requirement to scan the whole search space.

Proposition 5.1. *Consider two episodes,* α *and* β*, with the condition that* $\beta \sqsubseteq \alpha$*. If the episode* α *is frequent, then the episode* β *is also frequent. Conversely, if the episode* β *is infrequent, then the episode* α *is also infrequent.*

Proof. Considering that $\beta \sqsubseteq \alpha$, every occurrence of α in sequence S must include an occurrence of β in S. The opposite, however, could not always be true as an occurrence of β might have a continuation that doesn't always result in an occurrence of α.

As a result, $support(\alpha) \leq support(\beta)$, meaning that the number of occurrences of α is at most equal to the number of occurrences of β. Thus, it follows that $support(\alpha) \geq minsup$ if α is frequent. That being said, $support(\beta) \geq minsup$ since $support(\alpha) \leq support(\beta)$. In the same way, we show that α is uncommon if β is. □

Now, let us describe how **Algorithm 3** is carried out. It first scans at the event sequence S to identify the frequently occurring episodes of a size 1. The time interval $[t_k, t_k]$ is included in the

set no_e of occurrences of e for each event e that takes place at time t_k in S (lines 4-6). The frequent episodes collection is formed by first evaluating the support of these single-event episodes (lines 7-9). Subsequently, the function repeatedly executes the *occurrence recognition* function to discover more frequent episodes using a depth-first search strategy. This function adds β to the end of α (lines 11–16) to search for occurrences of $\alpha \sqcup \beta$ given an arbitrary episode α and a one-size episode β. Proposition 5.1 implies that depth-first search continues only for nodes associated with frequent episodes (line 14). In other words, if an episode is infrequent, then no super-episodes will be frequent, which eliminates the related search subspace.

Algorithm 3: Mining Frequent Episodes

Input: *minsup* - minimum support threshold

Output: F - the set of all frequent serial episodes

2 $P = \{\}$, $F = \{\}$, $\alpha = \emptyset$

4 **for** *each individual event* $e \in E$ **do**

6 $\quad no_e = \{\}$

8 **for** $k = 1$ *to* n **do**

9 $\quad$ % scan the sequence S

11 $\quad e$ = the event found at time t_k

13 $\quad no_e = no_e \cup \{[t_k, t_k]\}$

15 **for** *each individual event* $e \in E$ **do**

17 $\quad$ **if** $|no_e| \geq minsup$ **then**

19 $\quad\quad P = P \cup \{(e)\}$

21 $F = P$

23 **for** *each individual episode* $\alpha \in F$ **do**

25 $\quad$ **for** *each individual episode* $\beta \in P$ **do**

27 $\quad\quad no_{\alpha \sqcup \beta} = Occurrence\_Recognition(\alpha, \beta)$

29 $\quad\quad$ **if** $|no_{\alpha \sqcup \beta}| \geq minsup$ **then**

31 $\quad\quad\quad F = F \cup \{\alpha \sqcup \beta\}$

33 $\quad\quad\quad \alpha = \alpha \sqcup \beta$

35 **return** F

Algorithm 4: Occurrence Recognition

Input: episode α - an episode to grow

episode β - a single event episode to grow α by.

Output: $no_{\alpha \sqcup \beta}$ - set of non-overlapped occurrences of new episode $\alpha \sqcup \beta$

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

The details of the *occurrence recognition* function are presented in **Algorithm 4.** This algorithm is inspired from [72] was used. Nevertheless, the suggested technique eliminates the requirement for several sequence scans to obtain a set of occurrences for new episodes because of the peculiarities of the environment. An episode to grow, α, and a single-event episode, β, which is utilized to grow α, constitute the input. The results include occurrences of the recently created episode $\alpha \sqcup \beta$. The algorithm finds an occurrence O_j in no_β that starts after the end of O_i for each occurrence O_i of α in set no_α. Specifically, $O_j.start > O_i.end$, where $O_j.start$ is the start time of the j^{th} β occurrence, and $O_i.end$ is the end time of the i^{th} α occurrence. Then, for every $\alpha \sqcup \beta$ (lines 1–5), we create a new instance $[O_i.start, O_j.end]$. The function removes any instances of α that coincide with the new occurrence of the new episode (lines 6-7) in order to handle the problem of overlapping occurrences.

4.4.2 Extracting episode rules

The process then proceeds to finding all important episode rules in the form $\beta \Rightarrow \alpha$, where β is the prefix for α and α, β are two common episodes with non-overlapping frequencies. Recall that rules with a confidence level of at least the minimal confidence threshold $minconf$ are considered legitimate. Equation 4.1 provides the formula used to calculate the confidence level of a rule. An enormous search space is produced when all the possible combinations of

α and β are explored to generate an episode rule. Two methods are used to reduce this search space: (1) only the prefixes of a given episode α are considered as possible antecedents of that rule when that episode is the consequent of that rule. The definition of an episode rule includes this case. (2) It is shown that for a rule consequent α, some prefixes should not be considered as possible antecedents of this rule, as they would inevitably lead to incorrect rules. This is guaranteed by applying the following anti-monotony property (see Proposition 5.2):

Proposition 5.2. *Suppose α and β are two frequent episodes with β as a prefix of α. If the rule $\beta \Rightarrow \alpha$ is invalid, then (1) the rule $\beta' \Rightarrow \alpha$ is invalid for every episode β' that is a prefix of β ($\beta' \sqsubseteq \beta$), and (2) $\beta \Rightarrow \alpha'$ is invalid for every episode α' where α is a prefix of α' ($\alpha \sqsubseteq \alpha'$).*

Proof. Let's start with the first part of the proposition. Assume that the rule $\beta \Rightarrow \alpha$ is false for two frequent episodes α and β, where $\beta \sqsubseteq \alpha$. Now, let $\beta' \sqsubseteq \beta$. We obtain $support(\beta') \geq support(\beta)$ from Proposition 5.1. Thus, it follows that: $support(\beta') \geq support(\beta)$. It follows that: $conf(\beta' \Rightarrow \alpha) = \frac{support(\alpha)}{support(\beta')} \leq conf(\beta \Rightarrow \alpha) = \frac{support(\alpha)}{support(\beta)}$. However, while $\beta \rightarrow \alpha$ is invalid we have: $\frac{support(\alpha)}{support(\beta)} \leq minconf$. As a result, $\frac{support(\alpha)}{support(\beta')} \leq minconf$, indicating that $\beta' \Rightarrow \alpha$ is not true.In a similar manner, the proposition's second element may be shown.**Algorithm 5**, which is based on Proposition 5.2, takes an input sequence S and produces all of the acceptable episode rules. □

The input for the algorithm includes an event sequence S over a set of event types E, along with user-defined frequency thresholds *minsup* and confidence threshold $minconf$. The result is the complete collection of valid episode rules in the form of $\beta \rightarrow \alpha$. Hereafter, for an episode α with a length of l, we use $pred(\alpha)$ to represent its predecessor, which is the longest prefix of α, specifically the prefix of length $l-1$.

Using *minsup* as the minimal support threshold, the algorithm first extracts all frequent episodes (**Algorithm 3**). The program then produces all the valid rules for every frequent episode α, where α is the consequent. This technique computes the confidence of $\beta \Rightarrow \alpha$, starting with $\beta = pred(\alpha)$ as a possible antecedent. The algorithm moves on to the next shorter prefix if this rule is valid and is added to the collection of output rules. When the program finds a prefix β for which $\beta \Rightarrow \alpha$ is invalid, it stops searching for other prefixes contained in α. According to Proposition 5.2, any additional prefix β' would also result in an invalid rule.

Algorithm 5: Extracting Episode rules

Input: S: sequence of events on E (event types set), $minsup$: support threshold, $minconf$: confidence threshold

Output: R: complete set of episode rules.

```
2  F = FrequentEpisodes(minsup)
4  R = ∅
6  for each episode α ∈ F do
8      stop = false
10     β = pred(α)/* returns the predecessor of episode α*/
12     while not stop and β ≠ NULL do
14         if |no_α| / |no_β| ≥ minconf then
16             R = R ∪ {β ⇒ α}
18             β = pred(β)
19         else
21             stop = true
23 return R
```

4.5 Experimental results

As the form of episode rules proposed in this paper is novel, there are no existing works employing the same form for comparison with the performance of our algorithm. Consequently, the objective of this section is to gain insights into various aspects of our algorithm's behavior. The experimental results presented in this section are derived from two experiments conducted on synthetically generated sequences. The datasets are available on GitHub [112]. To create these sequences randomly, we specify: (1) the length of the sequence, indicating the number of events it contains, (2) the number of event types, and (3) the maximum duration between two consecutive events. We categorize the sequences into three types: large, medium, and small. Further details about the utilized sequences are provided in Table 4.1. All experiments are performed on a PC with 1.7GHZ Intel i5-4210U CPU, 8 GB memory running Linux environment.

Table 4.1: Details on the sequences that are used

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

4.5.1 Phase I: Mining frequent episodes

In this study, we conducted a comparison between the NONEPI algorithm proposed in this research and three established frequent episode mining algorithms. These algorithms are implemented in Java and are accessible in the SPMF library[1]. The algorithms under consideration include MINEPI, which employs a breadth-first approach with minimal occurrence-based frequency, as well as MINEPI+ and EMMA, both of which are depth-first algorithms utilizing head frequency.

The comparison involves examining the number of discovered frequent episodes, the maximum size of identified frequent episodes, and the execution time variation based on the support threshold. The average values for each type of sequence are computed from three different sequences (refer to Fig. 4.2).

For large, medium and short-sized sequences, respectively, Figures 4.2(a), (d) and (g) show how the number of recognized frequent episodes varies according to the varyiation of *minsup*. We find that, in comparison to the other algorithms, the number of often generated episodes in NONEPI responds more quickly to shifts in the support criterion. In the latter case, over a number of successive values of *minsup*, the number of created frequent episodes tends to stabilize. It is also noteworthy that NONEPI produces fewer frequent episodes for medium and large sequences than other algorithms. This can be explained by the fact that episodes with no overlap are less common than those with an overlap when a sequence becomes larger.

The effects of support threshold *minsup* on the maximal size of frequent episodes in sequences of large, medium, and short sizes are shown in Figures 4.2(b), (e) and (h), respectively.

[1]SPMF Homepage, http://www.philippe-fournier-viger.com/spmf/

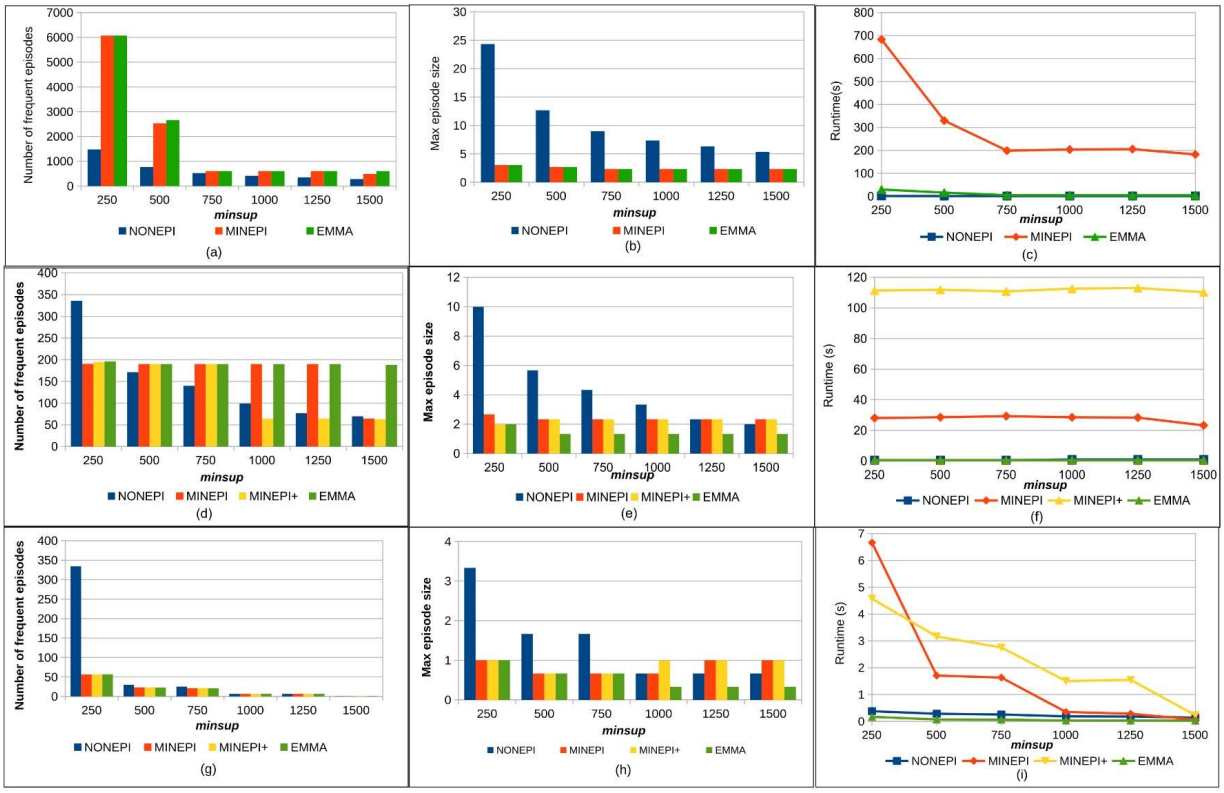


Figure 4.2: The effect of *minsup* on the amount of frequent episodes, the largest episode size, and the time of execution for large, medium, and short sequences.

Note that the maximum number of episodes found using NONEPI diminishes as *minsup* increases. However, regardless of the support threshold, the maximal size of episodes detected by MINEPI and MINEPI+ remains comparatively constant. This implies that in contrast to the other algorithms, the maximum size of frequent episodes in NONEPI is more sensitive to shifts in the support threshold. NONEPI's evaluation of independent events is the cause. More specifically, than overlapping occurrences or occurrences of shorter episodes, larger episodes naturally produce vast and non-overlapping occurrences, which occur less often. Large episodes that occur often at lower support thresholds may thus become less common as the threshold for support rises.

The runtime comparison of the methods for various support thresholds in sequences of large, medium, and short sizes is shown in Figures 4.2(c), (f) and (i), respectively. It is evident that NONEPI and EMMA have significantly shorter runtimes than the other two techniques. Among all the examined algorithms, MINEPI+ performs the worst for all types of sequences. Figures

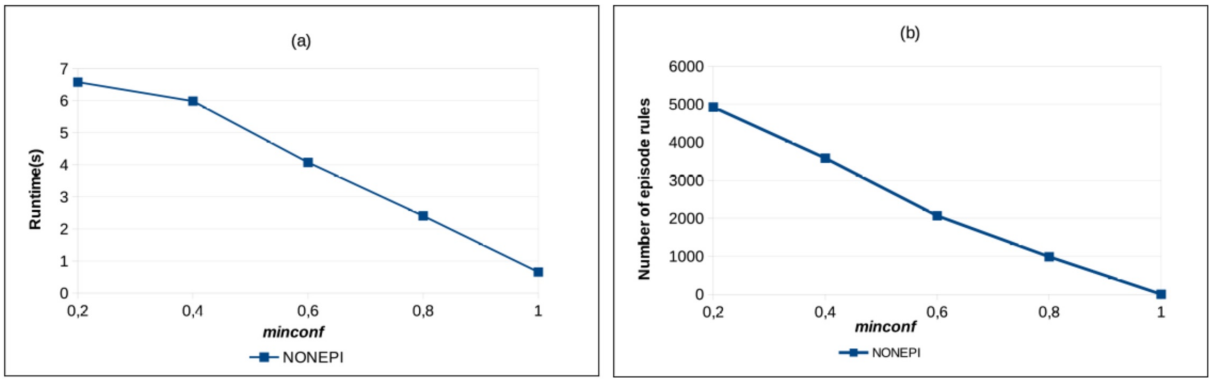


Figure 4.3: Influence of *minconf* on (a) execution time and (b) number of episode rules

(a), (b) and (c) of the large sequences do not show MINEPI+ because they fail to end before 800 s for large sequences. The reason for NONEPI's superior performance is that, unlike MINEPI and MINEPI+, which repeatedly scan the event sequence to identify the occurrences of the most recent episodes using the window width, NONEPI only scans the sequence once to extract the timestamps of individual event episodes and then joins them to obtain larger episode occurrences. This further demonstrates the effectiveness of the NONEPI pruning method, which is based on the anti-monotony property stated in Proposition 5.2.

4.5.2 Phase II: Mining episode rules

The proposed NONEPI for episode rule generation is demonstrated in this experiment. In accordance with the minimal confidence level, it displays changes in the execution time and the quantity of made rules. In order to concentrate on the confidence impact, we have set the support threshold to $minsup = 100$ and shown the average values of the number of created rules and execution time that were found in the nine tested sequences.

Figure 3(a) illustrates the impact of the confidence threshold (*minconf*) on the execution time, while Figure 3(b) shows its influence on the number of mined episode rules. With an increase in the confidence threshold, there is a gradual reduction in the number of rules, indicating that valid episode rules generated by NONEPI typically have high confidence values. Conversely, as the confidence threshold rises, the runtime experiences a rapid decrease. This suggests that obtaining the primary valid episode rules is achievable even with a relatively high confidence threshold, ensuring a shorter runtime.

4.6 Conclusion

This chapter introduced an effective method known as NONEPI and suggested a new class of episode rules based on the non-overlapped occurrence-based support. We have demonstrated that our episode rules enjoy the anti-monotonic property, which enables us to suggest a pruning strategy: Once we identify the largest prefix of α, let's say β, such that $\beta \Rightarrow \alpha$ is invalid, we cease generating episode rules with that episode α as a consequence. Future work on NONEPI might take numerous directions including: (1) increasing its use to mining episode rules with partially ordered events or to analyze uncertain data using a probabilistic model. (2) Consider event sequences that are more intricate, such as streams, and (3) develop a sequence prediction model based on our algorithm.

Chapter 5

A new approach for extracting episode rules in complex sequences under distinct frequency

There are several real-world applications where multiple events might happen at the same time, leading to the so-called complex sequence of events. Fortunately, many studies have proposed new algorithms for treating complex sequences. Within this chapter, we introduce a novel algorithm named Episode Mining based on Distinct Occurrences (EMDO) to find frequent episodes in a complex event sequence by using a definition of frequency that uses distinct occurrences. Additionally, we present an effective algorithm, called Episode rules mining under distinct occurrences with pruning (EMDO-P), for extracting episode rules from complex event sequences based on the distinct frequency definition.

5.1 Introduction

This chapter introduces a novel approach for discovering frequent episodes and episode rules using a another definition of occurrences called *distinct occurrences*. There are two main stages in this approach: the first stage focuses on identifying frequent episodes, while the second stage identifies valid episode rules in accordance with the proposed guidelines mentioned previously. Notice that, in this chapter, the definitions of "event", "event sequence" and "episode" introduced in Chapter 2.5 still hold. However, the definition of frequency is here determined by

episode's "distinct occurrences". The new proposed format of episode rules and the confidence of such rules will be thoroughly examined.

5.2 Preliminaries and problem definition

Before presenting the details of our novel method for episode and episode rules mining, we start by defining all necessary concepts used by the method. Next, we define the problem that we address in this chapter: mining frequent episodes as well as the interesting episode rules from a sequence by counting the distinct occurrences of episodes to measure their frequency.

Definition 6 (Complex Sequence of Events)**.** *Let E be as event types set, a complex sequence $S = \langle (X_1, y_1), (X_2, y_2), ..., (X_n, y_n) \rangle$ is an ordered set of couples (X_i, y_i), where $X_i \subseteq E$ is a subset of event types and y_i the occurrence time of the events of X_i in the sequence S.*

The approach that we present in this work uses complex sequences as input. For the sake of simplicity, we use throughout he chapter the word sequence to refer to any complex sequence.

Fig. 5.1 is an illustration of a complex sequence of 7 event sets on 5 distinct event types of the set $E = \{l, m, n, o, p\}$. The time interval begins at $y_1 = 8$ and ends at $y_7 = 14$. The example might be a representation of a server's logs where events are logged concurrently. It is evident to us that the server records various events, such as requests to open a link connecting a client to the host on certain port, for every timestamp that indicates an activity. Here, many computers may be aiming for the same port. In order to clearly build a complex event sequence, the system ought to track any new requests on any particular channel at a time.

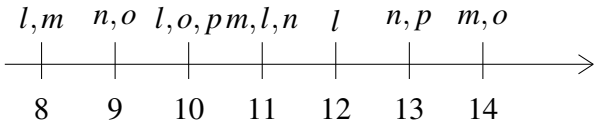


Figure 5.1: A complex sequence of events

Definition 7 (Episode occurrences, Distinct occurrences)**.** *Let S be a sequence S and $\gamma = A_1 A_2 \ldots A_n$ be an episode.*

- *In the sequence S, an episode occurrence of α is represented by an integers array $O = [y_1\ y_2\ \ldots\ y_n]$, in which every y_i represents the timestamp of occurrence of the i^{th} event of γ, i.e., A_i arises at y_i in the sequence.*

- *Let $O = [y_1\ y_2\ \ldots\ y_n]$ and $O' = [y'_1\ y'_2\ \ldots\ y'_n]$ be two occurrences. O and O' are considered* ***distinct*** iff, $\forall\ j, i \leq\ |\gamma|$, $y_i \neq y'_j$ *in which* $y_i \in O$ *and* $y'_j \in O'$

We can now define the different occurrences maximal set of an episode (with respect to set inclusion) as follows:

Definition 8 (Distinct occurrences maximum set)**.** *Let γ be an episode and consider D as its distinct occurrences set. For D to be a distinct occurrences maximum set of γ, it must satisfy that, for every other distinct occurrences set D', $|D| \geq |D'|$. We use the notation $do(gamma)$ to express that set.*

Consider the example sequence in Fig. 5.1 and an episode $\gamma = l\ m\ n$. $do(\gamma)$ = {[8 8 9],[10 11 11],[12 14 13]} is the distinct occurrences maximum set of γ.

Now, let us turn to the notion of support of an episode under the distinct frequency. Consider a sequence, the aim is to find every episode whose support, i.e. frequency is exceeds a minimum threshold *minsup*. The literature contains several frequency definitions, each of which conveys some concept of how frequently this episode happens in the sequence. Under distinct occurrence-based frequency, the support of an episode simply corresponds to the greatest number of its separate occurrences in the sequence. Therefore, an episode is considered as distinct occurrences frequent episode is its support exceeds the given threshold.

Definition 9 (Frequent episode, Episode support)**.** *Let S be a sequence and γ be an episode:*

- *Using a definition that bases on distinct occurrences, γ's support (expressed by sup(γ)) is the number of its distinct occurrences: $sup(\gamma) = |do(\gamma)|$ where $do(\gamma)$ is the distinct occurrences maximum set.*

- *If $sup(\gamma) \geq$ minsup, then γ is deemed as a "distinct occurrence frequent episode".*

Let us consider the example of the sequence depicted in Figure 5.1 and take $minsup = 3$. With respect to *minsup*, we derive the distinct occurrence-based frequent episodes. As a starting point, we seek for episodes of size 1. Consider the episode $\gamma = l$, its timestamps resultant set is simply $do(\gamma) = \{[8], [10], [11], [12], [14}\}$; thus, its support is $sup(\gamma) = 4$ .Then, we can find the maximum set of distinct occurrences for the rest of episodes of size 1, i.e, for event types $E = \{m,\ n,\ o\}$. Then, Definition 7 is used to join the occurrences of every pair of episodes

which leads to larger episodes. Table 5.1 shows the distinct occurrence-based frequent episodes with $minsup = 3$.

Table 5.1: Episodes deemed frequent with "$minsup = 3$"

γ	Times arrays	Sup(γ)
l	{[8], [10], [11], [12], [14]}	4
m	{[8], [11], [14]}	3
n	{[9], [11], [13]}	3
o	{[9], [10], [14]}	3
l m	{[8 8], [10 11], [12 14]}	3
l m n	{[8 8 9], [10 11 11], [12 14 13]}	3
l m n o	{[8 8 9 9], [10 11 11 10], [12 14 13 14]}	3
l n	{[8 9], [10 11], [12 13]}	3
l n o	{[8 9 9], [10 10 10], [11 14 14]}	3
l o	{[8 9], [10 10], [11 14]}	3
m n	{[8 9], [11 11], [14 13]}	3
m n o	{[8 9 9], [11 11 10], [14 13 14]}	3
m o	{[8 9], [11 10], [14 14]}	3
n o	{[9 9], [11 10], [13 14]}	3

We extend our methodology to extract important episode rules . As discussed in previous sections, there are many works to mine interesting rules. We propose a new set of rules in the format $\gamma \rightarrow \delta$ so that γ and δ are distinct occurrence-based frequent episodes. The precise definition of a rule's confidence varies based on the specific method employed and the structure of the rule itself.

Definition 10 (Episode Rule)**.** *An episode rule is a statement that asserts the relationship between two frequent episodes, α and β, under the condition of distinct occurrence-based frequency.*

Previously, it was noted that an episode rule unveils crucial connections between frequent episodes. Specifically, the significance of this rule under the final definition lies in the fact that if episode γ happens within a given sequence, another frequent episode δ will be triggered. Given that the approach is employing episodes with empty order, every event's occurrence of γ may initiate δ. In order to convey that δ is somehow influenced by γ, it is necessary that the beginning (resp. ending) of δ must occur subsequent to the beginning (or ending) of

α. Our latest episode rules take into account various other works, such as those that mine partially ordered episodes and rules based on a frequency based on non-overlapped occurrence, as demonstrated in [67] [68] as well as serial episodes and episodes rules as shown in [43].

Definition 11 (Occurrence of Episode Rule)**.** *Given a rule $\gamma \Rightarrow \delta$. Let γ_i be an occurrence of episode γ and δ_j be an occurrence of episode δ ($\gamma_i \in do(\gamma)$ and $\delta_i \in do(\delta)$). The vector $O = [t_{\gamma_1} \ldots t_{\gamma_n} t_{\delta_1} \ldots t_{\delta_m}]$ is an occurrence of the rule $\gamma \Rightarrow \delta$. The rule's occurrence O validity is determined by two conditions being met: the starting time of the episode, $T_s(\gamma_i)$, is less than the starting time of the episode, $T_s(\delta_j)$, and the ending time of the episode, $T_e(\gamma_i)$, is less than the ending time of the episode, $T_e(\delta_j)$. The functions T_s and T_e take the episode's occurrence as input and return its starting and ending times, respectively. $occER(\gamma \Rightarrow \delta)$ represents the set of all valid occurrences of the episode rule $\gamma \Rightarrow \delta$.*

The calculation of the support for an episode rule, represented by $\alpha \Rightarrow \beta$, involves tallying the total number of valid occurrences in the sequence, while simultaneously considering the occurrence of episode rules.

Definition 12. *(Support of a Rule) Given a rule $\gamma \Rightarrow \delta$. Its support is represented by $supER(\gamma \Rightarrow \delta)$ and it is defined as the number of its valid occurrences with respect to Definition 11, i.e:*

$$supER(\gamma \Rightarrow \delta) = |occER(\gamma \Rightarrow \delta)|$$

We typically measure the confidence of an episode rule as the proportion of times the rule is supported by the data, relative to the support of its predicating event (antecedent).

Definition 13. *(Confidence of a Rule) the confidence of a rule $\gamma \Rightarrow \delta$, is expressed by $conf(\gamma \Rightarrow \delta)$, which is the ratio of the occurrences' number of the rule $\gamma \Rightarrow \delta$ to the support of γ. This formula is expressed as:*

$$conf(\gamma \Rightarrow \delta) = \frac{|occER(\gamma \Rightarrow \delta)|}{support(\gamma)}$$

In spite of the set of frequent episodes previously presented (see Table 5.1), it is possible to derive a set of rules in a straightforward way. Taking a threshold for the confidence $minconf = 50\%$ as an example, let $\gamma = l$ and $\beta = o$ be frequent episodes, every pattern is recognized by the distinct occurrences. As per Definition 11, it is easy to derive the occurrences of the rule

$ER = l \Rightarrow o$ as $occER(l \Rightarrow o) = \{[8,9],[10,14]\}$, and thus, the number of its valid occurrences is $suppER(a \Rightarrow d) = \|occER(a \Rightarrow d)\| = 2$. Consequently, the rule's confidence is computed according to Definition 13 as $conf(l \Rightarrow o) = \frac{\|occER(l \Rightarrow o)\|}{sup(\gamma)} = \frac{2}{4} = 0.5$. The following list (see Table 5.2) shows the set of rules extracted from the sequence of Fig. 5.1 based on a confidence threshold of $minconf = 0.5$.

Table 5.2: Valid episode rules with $minconf = 0.5$

Rule $\gamma \Rightarrow \delta$	Times vector	conf($\gamma \Rightarrow \delta$)
$l \Rightarrow o$	{[8 9], [10 14]}	0.5
$l \Rightarrow o\ n$	{[8 9 9], [10 13 14]}	0.5
$l \Rightarrow o\ n\ m$	{[8 10 11 11], [10 14 13 14]}	0.5
$l \Rightarrow o\ n\ m\ l$	{[8 10 11 11 10], [10 14 13 14 12]}	0.5
$l \Rightarrow n$	{[8 9], [10 11], [11 13]}	0.75
$l \Rightarrow n\ m$	{[8 11 11], [10 13 14]}	0.50
$l \Rightarrow n\ m\ l$	{[8 11 11 10], [10 13 14 12]}	0.50
$l \Rightarrow m$	{[8 11], [10 14]}	0.50
$l \Rightarrow m\ l$	{[8 11 10], [10 14 12]}	0.50
$l \Rightarrow l$	{[8 10], [10 11], [11 12]}	0.75
$o\ l \Rightarrow o$	{[9 8 10], [10 10 14]}	0.67
$o\ l \Rightarrow o\ n$	{[9 8 10 11], [10 10 14 13]}	0.67
$o\ l \Rightarrow o\ n\ m$	{[9 8 10 11 11], [10 10 14 13 14]}	0.67
$o\ l \Rightarrow o\ n\ m\ l$	{[9 8 10 11 11 10], [10 10 14 13 14 12]}	0.67

The chapter aims to tackle the issue of frequent episodes and episode rule mining using distinct occurrence-based frequency. In other words, for a given sequence, a threshold of support *minsup*, and threshold of confidence $minconf$, the suggested method involves two main tasks:

1. Identifying all frequent episodes, i.e., having a support value which is equal to or greater than *minsup*.

2. Extracting every rule $\gamma \Rightarrow \delta$ where both γ and δ are frequent episodes, such that its confidence at least equals $minconf$ ($conf(\gamma \Rightarrow \delta) \geq minconf$).

5.3 The proposed approach

The subsequent subsections introduce our novel method for discovering frequent episodes and episode rules based on distinct occurrences frequency. As stated above, the proposed algorithm proceeds in two main stages: the first stage identifies frequent episodes and involves the recognizing of distinct occurrences, while the second stage extracts all valid episode rules based on the specifications that we have previously presented.

5.3.1 Phase I: Distinct occurrence-based frequent episode discovery

The process begins by identifying distinct occurrences of episodes. The function named "*Mine_frequent_episodes*", outlined in **Algorithm 6**, first examines the sequence passed as input to identify single-event episodes. For every type $e \in E$ that occurs at timestamp t_k, t_k is inserted to the set do_e, which saves the occurrences of e. Afterward, the function eliminates single-event episodes that do not occur frequently (lines 1-9). The algorithm identifies bigger episodes by combining single-event with n-node episodes repeatedly using a backtracking search strategy. "*Distinct_Occurrence_Recognition*" (Algorithm 7) is utilized to determine the maximum set of distinct occurrences for each generated episode. To expedite the search process, the function responsible for generating distinct occurrence-based frequent episodes takes advantage of the anti-monotonicity property, thereby avoiding the need to explore the entire space.

Proposition 13.1. *Let θ and γ be two episodes that satisfy $\theta \sqsubseteq \gamma$. If γ is a frequent episode, then θ is frequent as well. Similarly, if episode θ is infrequent, episode γ is infrequent too.*

Proof. As $\theta \sqsubseteq \gamma$, every occurrence of γ in the sequence S must also contain an occurrence of θ. Therefore, $sup(\theta) \geq sup(\gamma)$, meaning that the number of occurrences of episode γ do not exceed to the number of occurrences of episode θ. Hence, since episode γ is frequent, i.e., $sup(\gamma) \leq sup(\theta)$, then episode θ is frequent, too. Thus, the distinct occurrence-based support exhibits anti-monotonicity.

The function "*Distinct_Occurrence_Recognition*" takes two episodes as inputs: an episode α to grow using a size 1 episode β. Its purpose is to find the maximum set of distinct occurrences (denoted: $do_{\alpha \sqcup \beta}$) which is its output (Algorithm 7). The suggested algorithm efficiently detects the timestamps of new episodes of size $n+1$ by analyzing the occurrences of episodes

of size n and a single-event episode, without the need for multiple scans of the input sequence that are often required by other FEM algorithms, which can be time and resource consuming.

The algorithm processes each $O_i \in do_\alpha$ by identifying a distinct occurrence of β and constructing an occurrence of bigger episodes formed as the union of them with a timestamps' vector that combines the timestamps of O_i from do_α and the timestamps of the identified occurrence of β from do_β.

After identifying the new occurrence, the algorithm compares it with all occurrences in set $do_{\alpha \sqcup \beta}$ to determine whether there is overlap. An occurrence from do_α or do_β is removed if there is an overlap. O_j will be removed from do_β if the time stamp of the event in β exists in vector $newocc.timestamps$. In such a case, O_i will be removed from do_α since one timestamp from $O_i.timestamps$ exists in $newocc.timestamps$. The function $exists(t_k, O_i)$ outputs $true$ if the value t_k appears in the vector $O_i.timestamps$; otherwise, it returns $false$. For every $O_i \in do_\alpha$, our approach obtains $O_j \in do_\beta$ from do_β and constructs an occurrence of $\alpha \sqcup \beta$ with timestamps from O_i (i.e., $O_i.timestamps$ from do_α) and O_j (i.e., $O_j.timestamps$ from do_β).

5.3.1.1 An illustrative example

Let $minsup = 3$ and consider the sequence depicted in Figure 5.1. For Algorithm 6, each E event type is considered as an individual episode (i.e., an episode with only one event). By scanning the complex event sequence, the method determines, for each episode, the number of its occurrences, along with its support according to definition 9 (see lines 1-14). Based on the support threshold, the method then removes infrequent single-event episodes (see lines 15-19). Based on the execution of lines 1-19, a set of frequent episodes P is shown in Table 5.3.

The use of a depth-first strategy involves creating a copy of frequently occurring episodes and then searching for larger ones. In order to determine whether an episode is frequent, the method counts its "distinct occurrences", expressed by $do_{\alpha \sqcup \beta}$. This process will be illustrated in the following example. In the next step, the algorithm checks that the episode's support matches the requirements. As soon as the episode matches the criteria, it is added to the frequent episodes set F, and the algorithm continues its search (lines 21-28). The execution of Algorithm 6 invokes Algorithm 7 to carry out the step of "*distinct occurrences recognition*".

Take two episodes $\alpha = l$ and $\beta = m$ where $do_\alpha = [[8], [10], [11], [14]]$ and $do_\beta = [[8], [11], [14]]$.

Algorithm 6: Mine_Frequent_episodes

Input: *minsup* - a minimum support threshold.
Output: 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
{% 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
end
for *(each individual event* $e \in E$*)* **do**
    if *(*$\|do_e\| \geq minsup$*)* **then**
        $P \Leftarrow P \cup \{(e)\}$;
    end
end
$F \Leftarrow P$;
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
    end
end
return F;

Table 5.3: Size 1 frequent episodes for $minsup = 3$

Episode	Occurrences	Support
l	$\{[8],[10],[11],[14]\}$	4
m	$\{[8],[11],[14]\}$	3
n	$\{[9],[11],[13]\}$	3
o	$\{[9],[10],[14]\}$	3

During each iteration in Algorithm 7, a new occurrence is created and its timestamps are initialized as the union of α and β occurrences. As previously stated, the occurrence of $\alpha \sqcup \beta$ will arise from the first timestamps $O_\alpha = [8]$ and $O_\beta = [8]$, such that $newocc_{\alpha \sqcup \beta}.timestamps = [8\ 10]$. For the next occurrence of $\alpha \sqcup \beta$, the algorithm is repeated for subsequent occurrences of $O_\alpha = [10]$ *and* $O_\beta = [11]$. The set $do_{\alpha \sqcup \beta}$ is where each occurrence of timestamps $[10] \cup [11] = [10\ 11]$ is stored. However, there is a rare instance in which an $\alpha \sqcup \beta$ occurrence fails to satisfy the requirements of definition 7 (line 11).

In such a situation, the function chooses which occurrence to overstep using a different method.

Thus, for the same values of α_i ($\alpha_i \in do_\alpha$), the method selects the next value of β_j ($\beta_j \in do_\beta$) iteratively in which the timestamp of the single event episode β is already present in any previous occurrence of $\alpha \sqcup \beta$ or not. If not, it retains β_j ($\beta_j \in do_\beta$) and evaluates its join with the next α_i because it surely does not meet the condition of Definition 7 for such an occurrence of $\alpha \sqcup \beta$.

Take two episodes, as an example, $\alpha = l$ and $\beta = m$. The first occurrence of $\alpha \sqcup \beta$ is $[10\ 11]$ because the method joins the first vectors $O_\alpha = [10]$ and $O_\beta = [11]$ which meet the requirement of a valid occurrence of $alpha \sqcup \beta$. In the next iteration, the timestamp $t = 11$ already exists in $[10\ 11]$, thus the subsequent combination of $O_\alpha = [11]$ and $O_\beta = [14]$ will not be valid.Therefore, the technique merely maintains the time stamp $O_\beta = [14]$ and goes to the following vector $O_\alpha = [12]$ and gives a valid occurrence of $\alpha \sqcup \beta$ which outputs: $do_{\alpha \sqcup \beta} = \{[8\ 8],[10\ 11],[12\ 14]\}$.

All frequent episodes are located when the algorithm ends. Table 5.1 showcases the concluding set of frequent episodes that were produced by Algorithm 6 for the given complex event sequence.

Algorithm 7: Distinct Occurrence Recognition

Input: episode α - an episode to grow

episode β - a single event episode to be used to grow α.

Output: $do_{\alpha \sqcup \beta}$ - the set of distinct occurrences of the new episode $\alpha \sqcup \beta$

2 $j \leftarrow 0$;

4 $i \leftarrow 0$;

6 **for** *(each* $O_i \in do_\alpha$*)* **do**

8 $found \leftarrow false$;

10 **for** *(each* $O_j \in do_\beta$ *and* $found = false$*)* **do**

12 $newocc.timestamps \leftarrow O_i.timestamps \cup O_j.timestamps$;

14 $stop \leftarrow$ **false**;

16 $k \leftarrow 1$;

18 **while** *(**not** stop **and** $k \leq |do_{\alpha \sqcup \beta}|$)* **do**

20 **if** *(*$newocc.timestamps \cap O_k.timestamps \neq \emptyset$*)* **then**

22 **if** *(*$exists(O_j.timestamps[1], O_k.timestamps) =$ ***true****)* **then**

24 remove O_j from do_β;

25 **else**

27 remove O_i from do_α;

28 **end**

30 $stop \leftarrow$ **true**;

31 **end**

33 $O_k \leftarrow O_{k+1}$;

34 **end**

36 **if** *(stop)* **then**

38 $found \leftarrow$ **true**;

39 **end**

41 $do_{\alpha \sqcup \beta} \leftarrow do_{\alpha \sqcup \beta} \cup newocc$;

42 **end**

43 **end**

45 **return** $do_{\alpha \sqcup \beta}$;

5.3.2 Phase II: Episode rules generation

The subsequent sections offer an expansion of the algorithm introduced in Section 5.3.1 to obtain every rule of the format $\gamma \rightarrow \delta$ in which both γ and δ are distinct occurrences of frequent episodes.

According to Algorithm 9, **"Extract_Episode_Rules"** defines the episode rule mining procedure. This method returns R the set of valid episode rules after receiving as inputs a complex sequence of events S as well as the thresholds for the support and confidence denoted by $minsup$ and $minconf$ respectively.

As starting point, The **Mine_Frequent_Episodes** function (Algorithm 6) is called by the algorithm to calculate the set of frequent episodes and subsequently, the set of frequent episodes is used to derive valid episode rules based on Definition 12. Indeed, using Definition 12, the **"Episode_Rule_Support"**'s function computes the rule's support in this process (see Algorithm 8).

Algorithm 8: "Episode Rule Support"

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

```
{% Inialization %}
ruleSupport ← 0;
i ← 1;
j ← 1;
while (i ≤ |do_α|) do
    stop ← false;
    while (j ≤ |do_β| and not stop) do
        for (each O_k in do_β s.t: j < k ≤ |do_β|) do
            if (start(O_i) < start(O_j) and end(O_i) < end(O_j)) then
                i ← i + 1;
                ruleSupport ← ruleSupport + 1;
                stop ← true;
            end
        end
        j ← k;
    end
end
return ruleSupport;
```

5.3.3 Discovering episode rules with pruning

Because all potential combinations of γ and δ are considered, the space for search of episode rules is large. A pruning strategy is used to reduce this issue. To be more precise, only the super-episodes of a specific single-event episode that arise from an episode rule are considered possible consequences of legitimate episode rules. The function **"Extract_Episode_Rules"**, which accepts a sequence of event sets S, threshold for support and confidence, $minsup$ and $minconf$ respectively, as inputs, facilitates the episode rule mining process. All valid episode rules set (denoted as R) is the output of the function. As indicated by Proposition 13.2, this is produced in the rule generation stage by employing the anti-monotony property.

Proposition 13.2. *Let γ and δ be two distinct occurrences of frequent episodes. For every δ' that satisfies $\delta \sqsubseteq \delta'$, if $\gamma \to \delta$ is not valid, then $\gamma \to \delta'$ is not valid too.*

Proof. To demonstrate that the confidence of a rule $\gamma \to \theta$ derived from a single event episode δ such that $\delta \sqsubseteq \theta$ will be invalid, one needs to utilize only the anti-monotonicity of the distinct occurrences-based support. Suppose that there are two episodes, $\delta \sqsubseteq \theta$, and that rule $\gamma \to \delta$ does not have support. Based on Proposition 13.1 (anti-monotonicity property), $sup(\delta) \geq sup(\theta)$. Hence,

$$conf(\gamma \to \delta) = \frac{\|occER(\gamma, \delta)\|}{sup(\gamma)} \geq conf(\gamma \to \theta) = \frac{\|occER(\gamma, \theta)\|}{sup(\gamma)}$$

. Although the rule $\gamma \to \delta$ is not valid, it follows that $conf(\gamma \to \delta) < minconf$. Consequently, the rule $\gamma \to \theta$ is also deemed invalid.

■

From the input complex event sequence, Algorithm 9 incorporates a pruning technique based on Proposition 13.2 to create all valid episode rules.

The algorithm initializes a variable R as the rules set to be generated as well as another one, named P that represents the size 1 frequent episodes, after the algorithm first establishes the frequent episodes set with respect to the threshold of support $minsup$. The method then combines large episodes with single-event episodes to determine the valid consequents set that form such rules in which the confidence exceeds threshold $minconf$ for every frequent episode α acting as the antecedent of rule $\alpha \to \beta$. The j^{th} single-event is used by the algorithm first

Algorithm 9: Extracting Episode rules with pruning

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

$F \leftarrow$ **FrequentEpisodes**$(minsup)$;
$R \leftarrow \emptyset$;
$P \leftarrow \{\gamma | \gamma \in F \wedge \|\gamma\| = 1\}$;
for *(each* α *in* F*)* **do**
    $j \leftarrow 0$;
    while *(*$j < |P|$*)* **do**
        $\beta \leftarrow P[j]$;
        $root \leftarrow \beta$;
        $k \leftarrow j$;
        $stop \leftarrow$ **false**;
        while *(****not*** $stop$*)* **do**
            if *(*$\beta \in F$*)* **then**
                $ruleSupport \leftarrow$ **EpisodeRuleSupport**(α, β);
                $conf \leftarrow \frac{ruleSupport}{|alpha.occ|}$;
                if *(*$conf >= minconf$*)* **then**
                    $R \leftarrow R \cup \{\alpha \rightarrow \beta\}$;
                    $root \leftarrow \beta$;
                else
                    $\beta \leftarrow root$;
                end
            else
                $\beta \leftarrow root$;
            end
            if *(*$k > |P|$*)* **then**
                $stop \leftarrow$ **true**;
            else
                $\beta = \beta \sqcup P[k]$;
                $k \leftarrow k + 1$;
            end
        end
        $j \leftarrow j + 1$;
    end
end
return R;

as a rule consequent. This method utilizes the episode *root* in line 8 to return and reverse it if no other possible super-episode of β develops as a consequent. For each episode β, the technique computes the support of the rule and its confidence using the **"Episode_Rule_Support"** algorithm. The method inserts episode rule $\alpha \Rightarrow \beta$ into the set of all valid rules and updates the *root* episode for the succeeding combination of consequents (lines 11–19) if the confidence exceeds the confidence threshold. If there are no more candidates for single-event episodes, the algorithm terminates; otherwise, it modifies episode β to increase the subsequent j^{th} episode from P (lines 20–23).

5.3.3.1 An illustrative example

The following paragraphs present a scenario to show how to utilize the technique. Algorithm 9 is used to find the set of all episode rules in the sequence S, as seen in Fig. 5.1 for $minsup = 3$ and $minconf = 50\%$. As previously said, the algorithm first determines the list of frequent episodes (see Table 5.1). Next, it takes the sequence S and builds a size 1 frequent episodes set (line 3), thus $P = \{l,\ m,\ n,\ o\}$. The technique then looks up for rules.

Consider the episode $\alpha = m$. The method identifies every rule for which $\alpha = l$ ($\alpha \in F$) is the antecedent. Next, it treats $\beta = m$ ($\beta \in P$) as the root of prospective consequents and the first consequent (lines 4–10), where $do_\alpha = \{[8],[10],[11],[12],[14]\}$ $do_\beta = \{[8],[11],[14]\}$. The rule's support of $ER = \alpha \rightarrow \beta$ (line 13) is then determined by the algorithm using the definitions 11 and 12.

The support is computed in this case as follows: Knowing that $do_\beta = \{[8],[11],[14]\}$ and $do_\alpha = \{[8],[10],[11],[14]\}$, for every occurrence $O_i \in do_\alpha$, Algorithm 8 is used to identify the occurrence $O_j \in_\beta$ such that $start(O_i) < start(O_j)$ and $end(O_i) < end(O_j)$. The episode rule $ER = \alpha \Rightarrow \beta$ has [8 11] as its first occurrence and cannot have [8 8] since $O_\alpha = [8]$ and $O_\beta = [8]$ do not satisfy the prior requirements. As a result, the technique advances to the $2^n d$ occurrence of β ($O_\beta = [11]$), confirming that the rule has occurred.

The technique then advances to subsequent time stamps of $O_\beta = [12]$ and $O_\alpha = [10]$ of β and α, respectively, producing vector [10 12], which deemed valid rule occurrence. The support of the rule $\alpha \rightarrow \beta$ is $supER(\alpha \rightarrow \beta) = \|occER(\alpha \rightarrow \beta)\| = 2$, and the set of occurrences of the rule is $ER - occ(\alpha \rightarrow \beta) = \{[8\ 11],[10\ 14]\}$.

Subsequently, Algorithm 9 proceeds with its computation of the confidence on line 14. The present episode β represents the root of prospective consequences if the confidence is higher than the $minconf$ threshold. In this case, the procedure is repeated and the current rule is deemed legitimate. It follows by being linked to the following episode $\gamma = n$ in the collection P. Notice that the root replaces the size 1 episode to join if the consequent is infrequent. The rule $l \Rightarrow \beta$ is valid for $minconf = 0.5$. Since $\beta = m$ is the root for subsequent consequents, it is paired with episode $\gamma = n$. Thus, $l \rightarrow m\ n$ is the rule for the following episode. Proposition 13.2 states that $\beta = m$ will be joined with $\delta = o$ and so on if the rule $l \rightarrow m\ n$ is invalid with regard to $minconf$. The technique derives the final valid rules set, which are presented in Table 5.2.

5.4 Experimental Results

Numerous tests were carried out on both generated and real datasets to demonstrate the effectiveness of our novel method for mining frequent episodes. The datasets are available of course on GitHub [113] Since no previous work has been done on frequent episode extraction from complex event sequences under distinct occurrence-based frequency definition, the results exclusively address the performance of our technique. 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.

5.4.1 Data generation

As previously stated we made use of many synthetic datasets. Three primary factors were used to produce these datasets at random: (i) the events number in the complex sequence (length), (ii) the number of different types of events, and (iii) The maximum number of event sets in a sequence per time stamp. Three categories of synthetic sequences were identified: short, medium, and long sequences. Real datasets are sourced from the SPMF datasets collection [1], and are essentially extensions of transaction databases. Since every item is seen as an event for this purpose, every transaction (itemset) is regarded as an event set in the intricate sequence. In addition, a transaction number is assigned to every event set in order to reflect the timestamp. The datasets are named *"OnlineRetail"* in the first instance, which has 541880

[1] SPMF Homepage, http://www.philippe-fournier-viger.com/spmf/

events and 2603 event types; *"FruitHut"* in the second, which contains 181970 events and 9390 event types; and *"Mushrooms"* in the third, which is a series of 8416 events.

5.4.2 Discovery of frequent episodes

The initial step of the proposed algorithm is to use Algorithm 6 to create a list of frequent episodes. The impact of the threshold of support on the number of frequent episodes and largest episode size on synthetic and real datasets is shown in Figures 5.2 and 5.3.

Using the suggested technique on synthetic and real datasets, it is evident from the extent to which the number of frequent episodes varies based on the support threshold values is crucial in determining the anti-monotony property of distinct occurrence-based frequency, which is highly effective at minimizing the space of search. Furthermore, when the minimum support value is increased, the size of the biggest frequent episode is remarkably reduced. For synthetic or real datasets, the more often big episodes occur, the lower the *minsup* values that indicate how often big episodes occur.

The results of the proposed EMDO algorithm applied to various *minsup* threshold values for generated data (short, medium, and large size) and real sequences (memory usage in megabytes) are shown in Figures 5.4 and 5.5, respectively.

In figures 5.4 and 5.5, the frequency of frequent episodes with regard to the *minsup* threshold decreased as the support increased. Moreover, with higher support values, the memory cost likewise drops down quickly decreases. This demonstrates, in particular, how well the anti-monotonicity propery works when applied to different occurrences-based frequency. According to this property, no episode's support is higher than that of any of its subepisodes. Consequently, the runtime and memory costs decrease with increasing support thresholds. Consequently, the outcomes demonstrated the effectiveness of the suggested strategy in terms of duration, memory use, frequency of episodes, and episode sizes.

5.4.3 Generation of episode rules

Formulating valid associations using the generated frequent episodes constitutes the $2^n d$ stage in EMDO. We compare the basic form of the developed EMDO algorithm with the modified version, EMDO-P, that uses the technique outlined in Sec. 5.3.3 to prune the space of

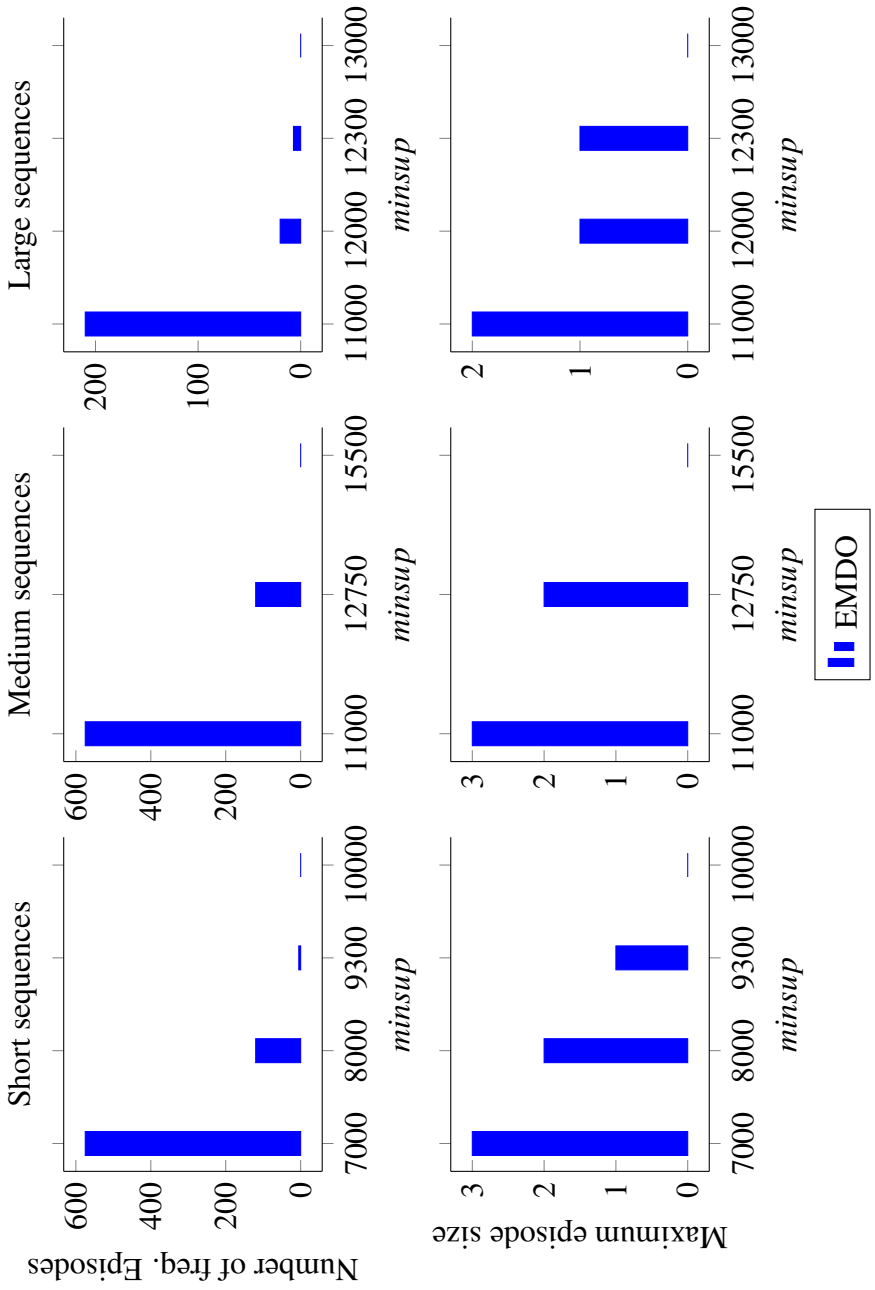


Figure 5.2: Effect of *minsup* on the biggest episode size and the frequent episodes count on synthetic datasets

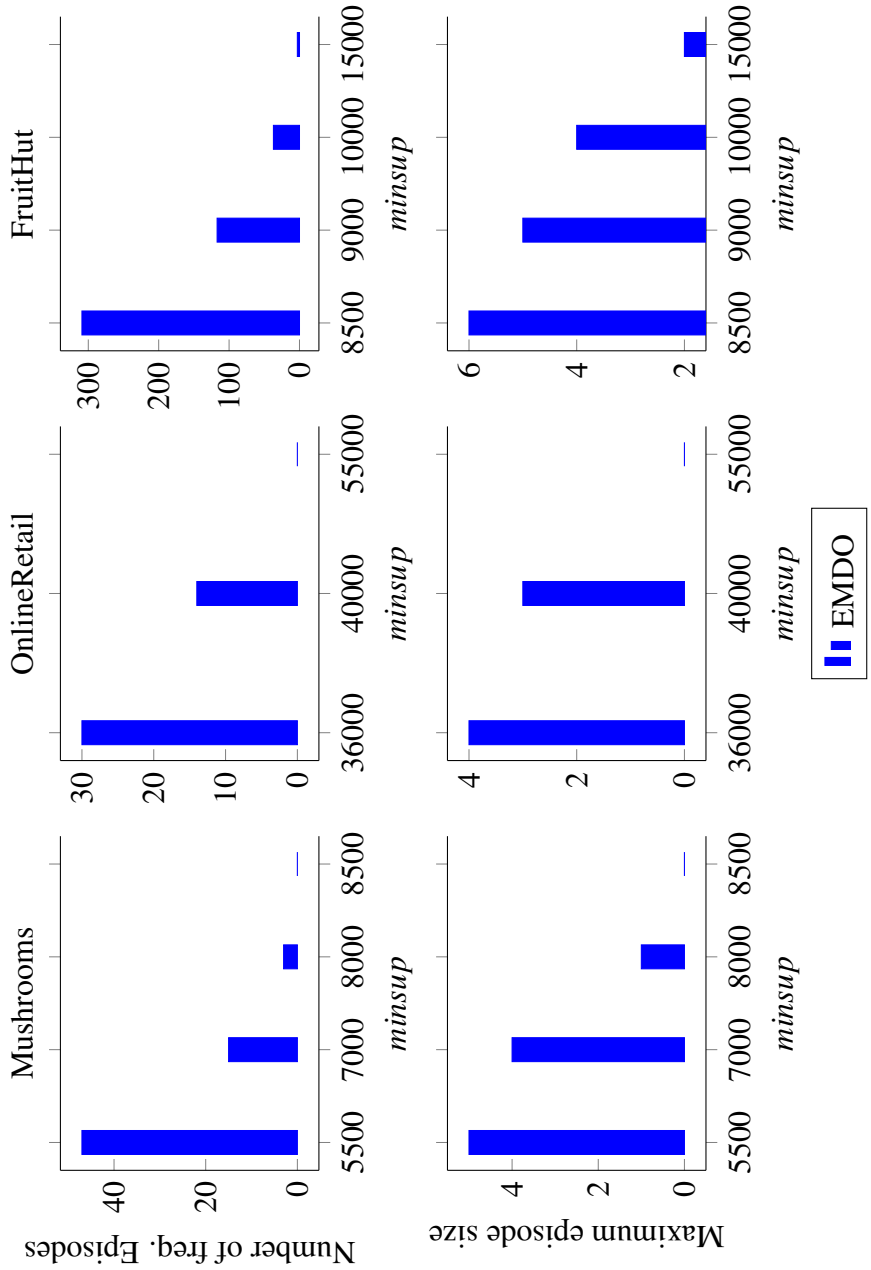


Figure 5.3: Effect of *minsup* on the biggest episode size and the size od the set of frequent episodes in real datasets

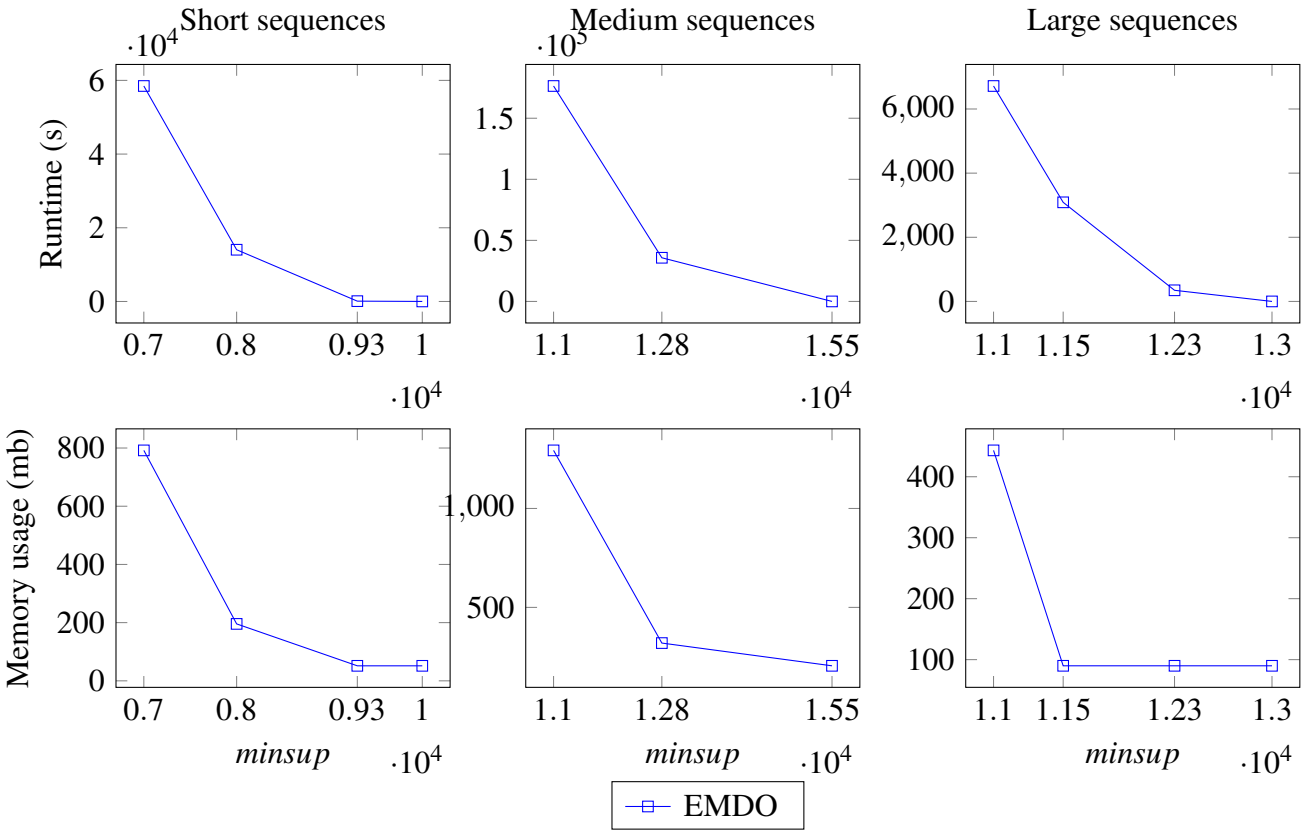


Figure 5.4: Effect of *minsup* on memory use and execution time for frequent episode discovery on synthetic data sets.

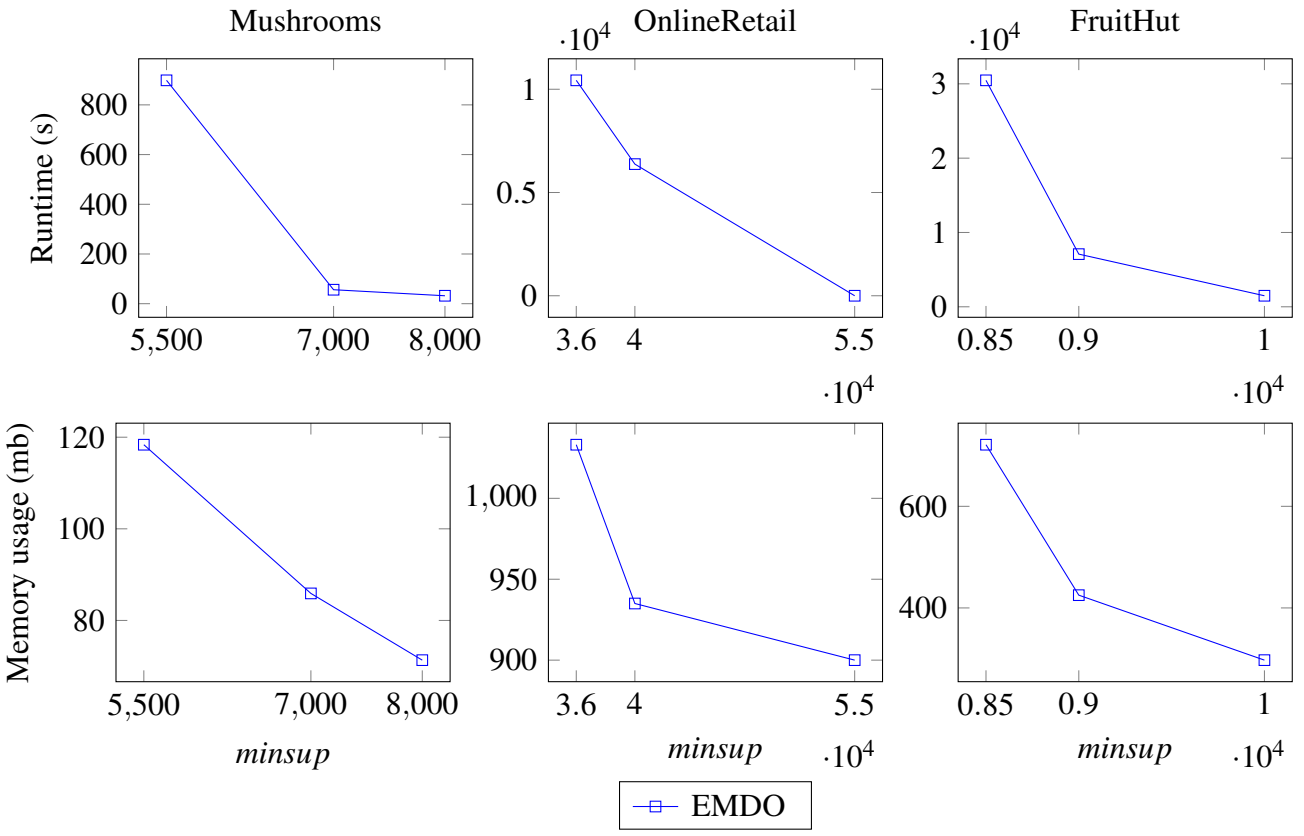


Figure 5.5: Effects of *minsup* on memory use and execution time on real data sets

Table 5.4: The values of support utilized in the rule generating process

Dataset type	Data labels	Value of threshold for the support
Synthetic datasets	Short dataset	1000
	Medium dataset	1750
	Large dataset	3000
Real datasets	Mushrooms	1050
	FruitHut	5000
	OnlineRetail	10000

search process, because no other algorithms use a support based on distinct occurrences to derive frequent parallel episodes and/or the rules set in complex sequences of events. The primary purpose of this experiment was to evaluate how effectively the proposed approach creates interesting rules. It is noticeable that although both versions of the algorithm provide identical results, meaning they produce the same episode rules, their respective performances vary, as explained in more detail below. Table 5.4 lists the minimum support values. Figures 5.6 and 5.7 illustrate the impact of the $minconf$ threshold on execution time (in seconds) and the consumption of memory (in megabytes) for the base technique EMDO and the version that uses the showed pruning strategy EMDO_P on generated and real datasets, respectively.

As predicted, the naive approach takes few memory space to construct pertinent rules for various confidence thresholds $minconf$ than the pruning algorithm. This insight is applicable to both synthetic and actual datasets. However, this difference in memory consumption is not cause for alarm because the higher memory cost is modest and will not affect the updated algorithm's performance.

The upgraded **EMDO_P** approach surpasses the basic **EMDO** version in with regard to the processing time for both generated and real datasets, especially when the confidence limit is raised. **EMDO_P**'s pruning approach eliminates large episodes that do not follow legitimate regulations. This benefit stems from our algorithm's new episode rule pruning method, which prevents the combination of several common episodes as opposed to the basic search technique.

5.4.4 A discussion of several identified episodes and rules

The **EMDO_P** method can extract relevant associations from sequential data and discover unexpected correlations between events. This was proved by applying it to the $FruitHut$ dataset, which resulted in a set of episode rules that were extremely significant. Table 5.5 presents an

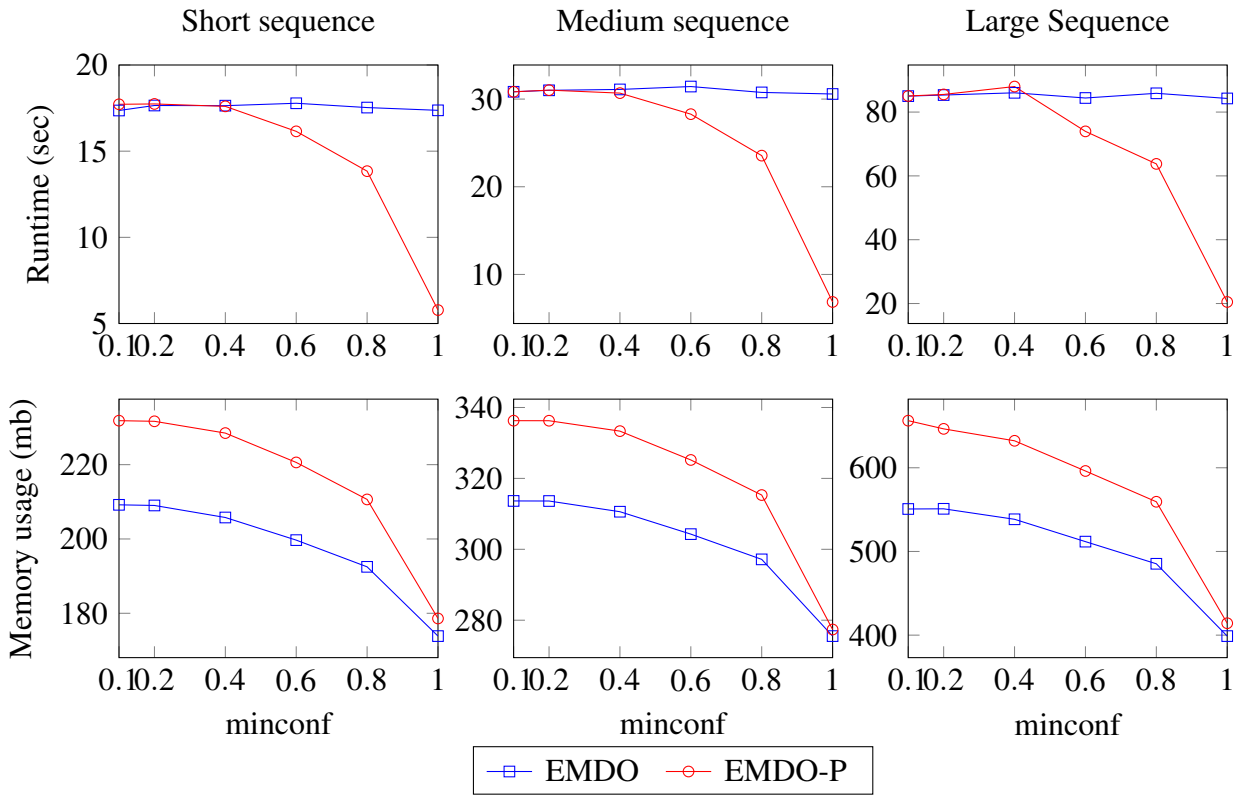


Figure 5.6: The impact of *minconf* on the memory and runtime used by rules formulation process on generated data

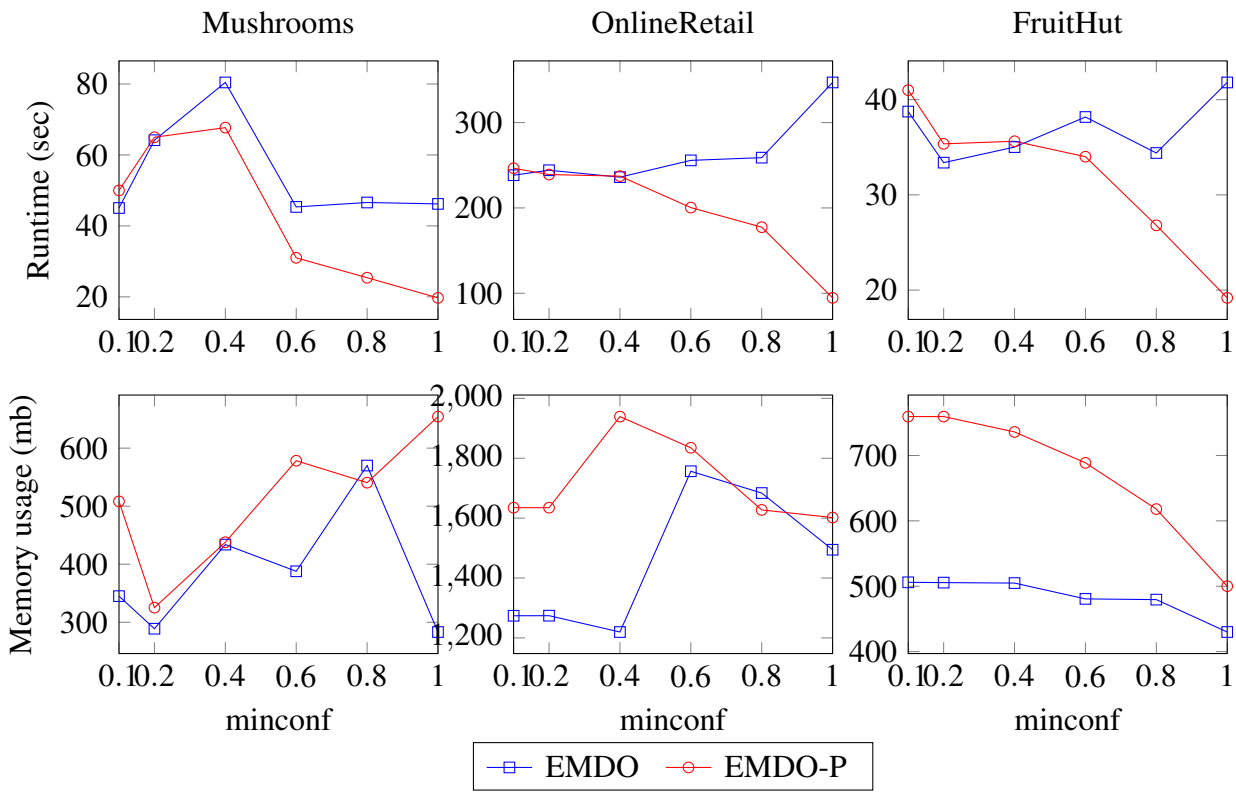


Figure 5.7: The impact of *minconf* on the memory and runtime used by rules formulation process on real data

Table 5.5: Patterns detected in the *FruitHut* dataset example.

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

example of the retrieved rules.

These criteria indicate strong customer-purchase connections. It is worth noting that the NONEPI [43] algorithm cannot produce those rules since it only generates rules in which the antecedent is a subepisode (predecessor) of the consequent. Fortunately, our technique may integrate all episodes to produce rules, not only those in which an antecedent is a subepisode of a consequent. Furthermore, the NONEPI algorithm's associations form may be created with the EMDO_P algorithm. For example, both NONEPI and EMDO_P created the rule "$Field\ Tomatoes \rightarrow Field\ Tomatoes, Banana\ Cavendish$" with a confidence level of $minconf = 50\%$. NONEPI can not create any of the rules provided in Table 5.5, save for the previously described one.

The significance of those rules lies in their ability to demonstrate the connections between goods that exist over time. For example, if a consumer purchases certain items in a certain sequence, they will also purchase other items of the antecedent of the rule, indicating their preferences or demands. Consequently, marketing plans based on promotions or suggestions can be created using these guidelines.

Table 5.6: 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 5.7: Information on several methods using the *FruitHut* dataset

	MINEPI+	EMDO	NONEPI	MINEPI
Candidate episodes number	26642	9959	1942	3993
frequent episodes number	992	3243	35	1244
Size of the biggest frequent episode size	8	6	2	13

We also compared the patterns discovered by "EMDO" and other discovering episodes methods, including the "MINEPI+", "NONEPI", and "MINEPI" techniques, on the real *FruitHut* dataset.

In order to determine the support of episodes, MINEPI+ employs a frequency definition known as the *head frequency*, whereas NONEPI mines *non-overlapped occurrences* frequent episodes and MINEPI uses the minimum occurrence-based frequency.

Table 5.6 compares the support values of frequent episodes found by all techniques are compared in Table 5.6. The table shows that the support determined by the proposed method is smaller than that determined by "MINEPI+", and larger than that determined by "NONEPI" and "MINEPI". Based on the fact that "EMDO" identifies a greater number of occurrences than "NONEPI" and "MINEPI", it is suggested that it may offer a more accurate representation and locate the frequently occurring episodes by exploring a smaller search area than "MINEPI+".

Because of the count of duplicate events with their timestamps in multiple windows of length k, the head frequency is larger than that of any other frequency specification. Consequently, for a given episode, the *head frequency* rises with the width of the window. The temporal frame that encompasses an episode's occurrence but lacks a suitable sub-window containing an alpha occurrence is known as the minimum occurrence of an episode. This constraint excludes non-minimal occurrences, because it states that any pair of occurrences can be distinct as long as they are minimal. Since there is no restriction on the existence of occurrences in such time intervals other than the fact that they do not share common events (timestamps), the distinct occurrence-based frequency is thus bigger than the minimal occurrence-based frequency. Finally, because it removes the majority of occurrences of each episode owing to the requirement that occurrences do not overlap, the lowest frequency for episodes is based on non-overlapping occurrences.

Additionally, Table 5.7 compares the count of candidate episodes, frequent episodes count,

and size of the biggest frequent episodes for every method. It is clear that EMDO has a higher ratio of frequent episodes to candidate episodes, demonstrating the effectiveness of our innovative strategy compared to existing algorithms. That is, on average, the EMDO has to investigate fewer possibilities to identify every valid pattern.

Overall, the tests demonstrate that EMDO is capable of identifying important occurrences in actual data. If other algorithms are used instead of EMDO, particularly those that employ occurrence definitions that share similar events, EMDO could offer a more accurate perspective because of its frequency function.

5.5 Conclusion

We discussed in this chapter episode rule mining from complex event sequences under the distinct occurrence-based frequency. We provided both the valid episode rules EMDO and its modified variant EMDO_P, as well as an effective depth-first technique to find common episodes. We have outlined every aspect of our strategy. To the best of our knowledge, this is the first effort that counts the frequency of parallel episodes from a complex event sequence using distinct occurrences.

Chapter 6

Mining frequent episodes and episode rules from uncertain complex sequences

In practical applications, it is often the case that data is not perfect due to a variety of reasons, including noisy sensors and multiple sources of information. As a result, one must be able to deal with such situations and extract useful knowledge from uncertain data. The traditional framework in episode mining aims to find significant subsequences among a set of candidates using an absolute frequency definition based on the occurrences of episodes in a simple sequence. However, this traditional approach is no longer valid for uncertain sequences and must be adapted. In this contribution, we focus on uncertain complex sequences, which are modeled using probability theory. We propose a new efficient algorithm called Uncertain Episode Mining based on Distinct Occurrences (UEMDO) based on the concept of expected support of an episode to discover "probabilistically" frequent parallel episodes and episode rules under distinct frequency from a complex uncertain sequence of events. We proposed also its extension to generate valid episode rules with pruning called Uncertain episode rules mining under distinct occurrences with pruning (UEMDO-P).

6.1 Episodes and Episode rules mining from uncertain complex sequences

For several reasons, real-life applications often produce noisy or uncertain data. Despite this uncertainty, data contains a significant amount of knowledge that can help us understand past or predict future events. Consequently, the uncertainty of data has also been studied in the context of frequent episode mining. Currently, there are only two existing techniques, PFSE and D-PFSE, which are designed to mine serial episodes from simple sequences under non-overlapping and distinct occurrence-based frequencies, respectively. However, to the best of our knowledge, there is no prior work that mines parallel episodes from complex uncertain sequences based on their expected support under a distinct occurrence-based frequency definition.

This chapter presents two algorithms for mining frequent episodes and episode rules in uncertain complex sequences, which are the main contributions of this research. Prior to describing these algorithms, we first provide an overview of the probabilistic context in which they operate. In the following subsection, we define the concepts of probabilistic event, uncertain sequence, and probabilistic occurrence, which are extensions of the concepts used in deterministic context for the probabilistic setting.

6.1.1 Preliminaries of probabilistic data and problem definition

Definition 14. *(Probabilistic Event) A probabilistic event is a triple* $ev = (e, p, t)$ *such that* $e \in E$ *is the event type and* $p \in [0,1]$ *is the existential probability of the event e at the timestamp t in the sequence S.*

Definition 15. *(Uncertain Event Sequence) Given a set E of event types. A complex probabilistic sequence is a collection of event sets* ε_i *where* $1 \leq i \leq |S|$ *and each* ε_i *is a set of probabilistic events denoted as:* $\varepsilon_i = \{(e_i^1, p_i^1, t_i), (e_i^2, p_i^2, t_i), ...(e_i^k, p_i^k, t_i)\}$ *where* $k = |\varepsilon_i|$.

For example, Figure 6.1 shows a complex probabilistic sequence of seven timestamps and five event types. Each event is associated with its existential probability at each timestamp. Here, $\varepsilon_2 = \{(c, 0.5, 2), (d, 0.5, 2)\}$ is the 2^{nd} event set of probabilistic complex sequence S. Note that, in the probabilistic setting, the definitions of notions of episodes, sub-episodes, occurrence of episodes, distinct occurrences, and maximal sets of distinct occurrences remain the same by

considering only events associated with non-zero existential probabilities.

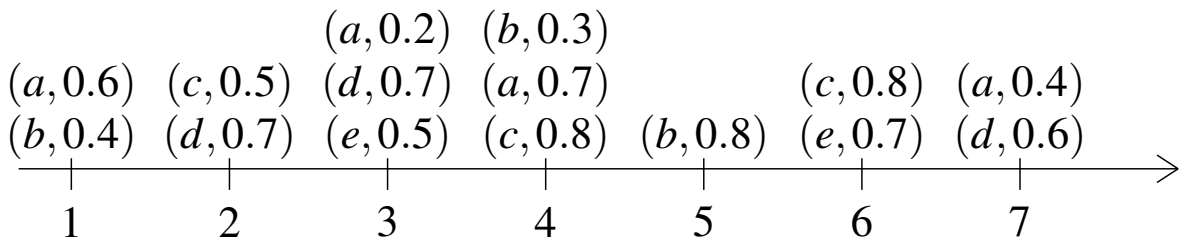


Figure 6.1: An example of an uncertain sequence

However, since the events in the sequence are uncertain, the occurrences of episodes are uncertain too. Hence, each occurrence of an episode α is associated with a probability value as follows:

Definition 16. *(Probabilistic Occurrence) Let $\alpha = A_1A_2\ldots A_n$ be an episode and $O = [t_1t_2\ldots t_n]$ be an occurrence of α in the sequence S. We denote by $Pr(A_i,t_i)$ the probability that event type A_i occurred at timestamp t_i in the sequence S, i.e, $(A_i,Pr(A_i,t_i),t_i)$ is a probabilistic event which occurs in S at timestamp t_i. Then, the probability of the occurrence O is given by*

$$p(O) = \prod_{1\leq i\leq n} Pr(A_i,t_i) \tag{6.1}$$

In the previous section about to episode mining from certain data, the events occur absolutely in the sequence. Consequently, in order to calculate the frequency at which an episode occurs within this sequence, we simply counted its occurrences (in that case, distinct occurrences). In the case of uncertain data, the occurrences of an episode are probabilistic; therefore, the support becomes dependent on the probability of its distinct occurrences. For this purpose, we introduce the notion of **expected support** of an episode α in a complex uncertain sequence S which is the sum of the probabilities of all distinct occurrences of α in S. Formally:

Definition 17. *(Expected-Support of an episode) Let S be complex uncertain sequence, α be an episode and $do_\alpha = \{O_1,O_2,\ldots,O_k\}$ be the set of probabilistic distinct occurrences of α. The expected-support of α in S is defined by:*

$$expsup(\alpha) = \sum_{O\in do_\alpha} p(O) \tag{6.2}$$

Given a support threshold *minsup*, an uncertain sequence S. The task is to mine all **frequent episodes** such that $expsup(\alpha) \geq minsup$.

The proposed UEMDO and UEMDO-P also capture episode rules in the probabilistic setting. Of course, since the occurrences of episodes are associated with probability values, the notion of support and confidence of an episode rule should be adapted to the probabilistic setting.

Definition 18. *(Episode Rule) let be α and β two episodes, an episode rule is an implication of the form: $\alpha \Rightarrow \beta$ such that α and β are two expected support-based frequent episodes under distinct occurrence frequency definition.*

Next, we introduce the notion of *episode rule occurrence* as follows:

Definition 19. *(Episode Rule Occurrence) Given an episode rule $\alpha \Rightarrow \beta$. A probabilistic occurrence of $\alpha \Rightarrow \beta$ is a vector $h = [t_\alpha^1, \ldots, t_\alpha^k, t_\beta^1, \ldots, t_\beta^m]$ such that:*

- *$[t_\alpha^1, \ldots, t_\alpha^k]$ is an occurrence of α.*
- *$[t_\beta^1, \ldots, t_\beta^m]$ is an occurrence of β.*
- *$t_\alpha^1 < t_\beta^1$ and $t_\alpha^k < t_\beta^m$ (i.e, the occurrence of α should strictly precede that of β).*

The set of all probabilistic occurrences of an episode rule $\alpha \Rightarrow \beta$ is denoted by ER-do$(\alpha \Rightarrow \beta)$

Definition 20. *(Expected-Support of Episode Rule) Given an episode rule $\alpha \Rightarrow \beta$. the expected-support of $\alpha \Rightarrow \beta$ is denoted by $expsup(\alpha \Rightarrow \beta)$ and it is defined as follows:*

$$expsup(\alpha \Rightarrow \beta) = \sum_{O \in ER-do(\alpha \Rightarrow \beta)} p(O) \tag{6.3}$$

Now, we can introduce how to measure the confidence of such an episode rule.

Definition 21. *(Confidence of an episode rule) Given an episode rule $\alpha \Rightarrow \beta$, The confidence of $\alpha \Rightarrow \beta$ is formulated as follows:*

$$conf(\alpha \Rightarrow \beta) = \frac{expsup(\alpha \Rightarrow \beta)}{expsup(\alpha)} \tag{6.4}$$

Then, the proposed problem is defined as follows: Consider (i) a complex sequence of probabilistic events S, (ii) a user-defined minimum support threshold *minsup* and (iii) a user-defined minimum confidence threshold $minconf$.

- The first research problem of UEMDO is the discovery of expected support-based *frequent episodes* from the sequence S i.e, episodes with an expected-support not less than *minsup* ($expsup(\alpha) \geq minsup$) based on the set of distinct occurrences.

- The second problem is to extract the set of valid expected support-based *frequent episode rules* $\alpha \rightarrow \beta$, i.e., rules $\alpha \Rightarrow \beta$ with a confidence which at least equals the threshold *minconf* ($conf(\alpha \rightarrow \beta) \geq minconf$).

For instance, consider a support threshold *minsup = 0.5* and the event sequence given in Fig. 6.1. Given episode $\alpha = a$, by definition 16, the set of distinct occurrences of α is $do_\alpha = \{[1],[3],[4],[7]\}$. Hence, the expected-support of α is $expsup(\alpha) = p(O_1) + p(O_2) + p(O_3) + p(O_4)$, which gives 1.9. All the frequent episodes and their expected-support values are listed in Table 6.1.

Episode	Occurrences	Expected-support
a	$\{[1],[3],[4],[7]\}$	1.9
b	$\{[1],[4],[5]\}$	1.5
c	$\{[2],[4],[6]\}$	2.1
d	$\{[2],[3],[7]\}$	2
e	$\{[3],[6]\}$	1.2
ab	$\{[1\ 1],[3\ 4],[5\ 7]\}$	0.62
ac	$\{[1\ 2],[3\ 4],[6\ 7]\}$	0.78
ad	$\{[1\ 2],[3\ 3],[4\ 7]\}$	0.98
ae	$\{[1\ 3],[4\ 6]\}$	0.79
bc	$\{[1\ 2],[4\ 4],[5\ 6]\}$	1.08
bd	$\{[1\ 2],[3\ 4],[5\ 7]\}$	0.97
ce	$\{[2\ 3],[4\ 6]\}$	0.81
cd	$\{[2\ 2],[3\ 4],[6\ 7]\}$	1.39
acd	$\{[1\ 2\ 2],[3\ 3\ 4],[6\ 7\ 7]\}$	0.514
bcd	$\{[1\ 2\ 2],[3\ 4\ 4],[5\ 6\ 7]\}$	0.692

Table 6.1: Frequent episodes for $minsup = 0.5$

Besides the frequent episodes set already demonstrated, the valid rules set can be computed. For instance, consider the previous frequent episodes set and that the confidence threshold is $minconf = 0.1$. Table 6.2 shows the set of valid episode rules that can be obtained based on definition 18 and definition 19, with their confidence values. Take the episode rule $ER = edc \rightarrow$

db as example. First of all, we must calculate the set of valid occurrences of that rule denoted by $ER-occ(edc \rightarrow db)$ according to definition 19. To calculate this set, we first start by the set of distinct occurrences of $\alpha = edc$ since it is the left hand side of the rule (the antecedent). For each $O_i \in do_\alpha$, we search for an occurrence $O_j \in \beta$ ($\beta = db$) such that they meet the condition mentioned in definition 19. Hence, we obtain the set of all valid occurrences of ER where $ER-occ(\alpha \rightarrow \beta) = \{[3\ 2\ 2\ 4\ 3]\}$. Then, we calculate the confidence $conf(\alpha \rightarrow \beta)$ according to the equation 6.4 which results in $conf(\alpha \rightarrow \beta) = 0.34$.

Episode rule	Episode rule occurrences	Confidence
$e \rightarrow e$	$\{[3\ 6]\}$	0.41
$e \rightarrow ed$	$\{[3\ 6\ 7]\}$	0.41
$e \rightarrow edc$	$\{[3\ 6\ 7\ 4]\}$	0.41
$e \rightarrow d$	$\{[3\ 7]\}$	0.41
$e \rightarrow db$	$\{[3\ 5\ 7]\}$	0.41
$e \rightarrow c$	$\{[3\ 4]\}$	0.41
$e \rightarrow b$	$\{[3\ 4]\}$	0.41
$e \rightarrow a$	$\{[3\ 4],\ [6\ 7]\}$	1
$ed \rightarrow e$	$\{[3\ 2\ 6]\}$	0.45
$ed \rightarrow ed$	$\{[3\ 2\ 6\ 7]\}$	0.45
$ed \rightarrow edc$	$\{[3\ 2\ 6\ 7\ 4]\}$	0.45
$ed \rightarrow d$	$\{[3\ 2\ 7]\}$	0.45
$ed \rightarrow db$	$\{[3\ 2\ 4\ 3]\}$	0.45
$ed \rightarrow c$	$\{[3\ 2\ 4]\}$	0.45
$ed \rightarrow b$	$\{[3\ 2\ 4]\}$	0.45
$ed \rightarrow a$	$\{[3\ 2\ 4]\}$	0.45
$edc \rightarrow e$	$\{[3\ 2\ 2\ 6]\}$	0.34
$edc \rightarrow ed$	$\{[3\ 2\ 2\ 6\ 7]\}$	0.34
$edc \rightarrow edc$	$\{[3\ 2\ 2\ 6\ 7\ 4]\}$	0.34
$edc \rightarrow d$	$\{[3\ 2\ 2\ 7]\}$	0.34
$edc \rightarrow db$	$\{[3\ 2\ 2\ 4\ 3]\}$	0.34
$edc \rightarrow c$	$\{[3\ 2\ 2\ 4]\}$	0.34
$edc \rightarrow b$	$\{[3\ 2\ 2\ 4]\}$	0.34
$edc \rightarrow a$	$\{[3\ 2\ 2\ 4]\}$	0.34

Table 6.2: Valid episode rules for $minconf = 0.1$

6.1.2 The proposed UEMDO algorithm for frequent episode discovery

In this section, we present the new approach that we propose for mining expected-support frequent episodes from complex probabilistic sequences under distinct-occurrence frequency. This process is divided into two main functions: *probabilistic distinct occurrence recognition* and *frequent episodes*.

Function *Frequent episodes*, given in **Algorithm** 10, computes the set of frequent episodes. The function takes as input a complex sequence of probabilistic events S and a support threshold *minsup*. Initially, the function extracts frequent episodes of size one. Here, the algorithm initializes, for each single-event episode, the set of probabilistic occurrences do_e, and then scans the sequence S to calculate each occurrence time t_k. Finally, the algorithm computes the **expected-support** according to Equation 6.2 given in **Definition** 17 (see line 1-9).

The function then applies a depth-first strategy to mine larger episodes and calculate the expected-support of each new episode using equation 6.2. If the expected-support exceeds the support threshold, the algorithm adds it to set F (see line 10-25).

In order to avoid considering all the search space, frequent episode generation using expected-support under distinct occurrence-based support utilizes the following anti-monotony property:

Proposition 21.1. *"Let α and β two episodes such that $\alpha \sqsubseteq \beta$, if 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. Given that $\alpha \sqsubseteq \beta$, it can be inferred that every occurrence of β in S must contain an occurrence of α. Thus, the maximum number of occurrences of episode β is equal to the number of occurrences of episode α. This can be formally expressed as $expsup(\alpha) \geq expsup(\beta)$. If episode β is frequent, i.e., if $expsup(\beta) \geq minsup$, then since $expsup(\beta) \leq expsup(\alpha)$, it follows that $expsup(\alpha) \geq minsup$. Hence, episode α is also frequent. As a result, the anti-monotony property holds for distinct occurrences-based support.

The function that determines the maximal set of probabilistic distinct occurrences is illustrated in **Algorithm** 11. This function takes as inputs an n-node episode and a single event episode, where the output is constituted of probabilistic occurrences, each of which is associated with a probability denoted by $O_i.prob$ where $O_i \in do_{\alpha \sqcup \beta}$. For each occurrence $O_i \in do_\alpha$, the function constructs a new occurrence of the new episode $\alpha \sqcup \beta$. Initially, the expected-support of the new episode is equal to $O_i.prob$ ($O_i \in do_\alpha$). Then, the function determines the occurrence of the single event episode O_j and updates the probability of the new occurrence of $\alpha \sqcup \beta$ by $O_j.prob$.

Algorithm 10: Frequent episodes

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

P= { }, F = { }, $\alpha = \emptyset$
for *each individual event* $e \in E$ **do**
  $do_e \leftarrow \{\}$
end
% scan the sequence S
for $k \leftarrow 1$ *to* n **do**
  % scan the event set found at time t_k
  for $j = 1$ *to* m **do**
    e_j = the event found at time t_k
    $do_{e_j} = do_{e_j} \cup \{t_k\}$
    $e_j.sup = e_j.sup + P_j^k$
  end
end
for *each individual event* $e \in E$ **do**
  if $e.sup \geq minsup$ **then**
    $P = P \cup \{(e)\}$
  end
end
$F = P$
for *each individual episode* $\alpha \in F$ **do**
  for *each individual episode* $\beta \in P$ **do**
    $do_{\alpha \sqcup \beta} = ProbabilisticDistinctOccurrenceRecognition(\alpha, \beta)$
    $expsup = 0$
    for O_k *in* $do_{\alpha \sqcup \beta}$ **do**
      $expsup = expsup + O_k.prob$
    end
    if $expsup \geq minsup$ **then**
      $\gamma = \alpha \sqcup \beta$
      $do_\gamma = do_{\alpha \sqcup \beta}$
      $\gamma.sup = expsup$
      $F = F \cup \{\gamma\}$
      $\alpha = \gamma$
    end
  end
end
return F

Algorithm 11: Probabilistic Distinct Occurrence Recognition

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

2 $j = 0, i = 0$
4 **for** *each* $O_i \in do_\alpha$ **do**
6 $\quad found \leftarrow false$
8 $\quad$ **for** *each* $O_j \in do_\beta$ *and* $found == false$ **do**
10 $\quad\quad newocc.timestamps \leftarrow O_i.timestamps \cup O_j.timestamps$
12 $\quad\quad newocc.prob \leftarrow O_i.prob$
14 $\quad\quad newocc.prob \leftarrow newocc.prob \times O_j.prob$
16 $\quad\quad stop \leftarrow$ **false**
18 $\quad\quad k \leftarrow 1$
20 $\quad\quad$ **while** ***not*** $stop$ ***and*** $k \leq |do_{\alpha \sqcup \beta}|$ **do**
22 $\quad\quad\quad$ **if** $newocc.timestamps \cap O_k.timestamps \neq \emptyset$ **then**
24 $\quad\quad\quad\quad$ **if** $exists(O_j.timestamps[1], O_k.timestamps) ==$ ***true*** **then**
26 $\quad\quad\quad\quad\quad$ remove O_j from do_β
27 $\quad\quad\quad\quad$ **else**
29 $\quad\quad\quad\quad\quad$ remove O_i from do_α
30 $\quad\quad\quad\quad$ **end**
32 $\quad\quad\quad\quad stop \leftarrow$ **true**
33 $\quad\quad\quad$ **end**
35 $\quad\quad\quad O_k \leftarrow O_{k+1}$
36 $\quad\quad$ **end**
38 $\quad\quad$ **if** $stop$ **then**
40 $\quad\quad\quad found \leftarrow$ **true**
41 $\quad\quad$ **end**
43 $\quad\quad do_{\alpha \sqcup \beta} \leftarrow do_{\alpha \sqcup \beta} \cup newocc$
44 $\quad$ **end**
45 **end**
47 **return** $do_{\alpha \sqcup \beta}$

6.1.2.1 Time complexity

The time complexity of UEDMO is next briefly analyzed. The algorithm has two functions. To compute the frequent episodes (see Algorithm 10), the algorithm scans the complex sequence and initializes the set of frequent episodes F with frequent episodes of size 1 (see line 4-12). Then, using only episodes of size = 1 (single events), the algorithm builds larger frequent episodes by repeatedly appending a single episode. Then, the algorithm finds their probabilistic episodes (see line 14-25) by calling the function for probabilistic distinct occurrences recognition function with two episodes. Given two episodes α ($|\alpha| > 1$) and β ($|\beta| = 1$), for each occurrence of α, the algorithm 11 tries to find an occurrence of β such that the two occurrences

meet Definition16. Hence, the algorithm stops the search when it finds a valid occurrence of β and repeats the process with other occurrences. Therefore, the complexity of the probabilistic occurrences search is equal to $O(|do_\alpha|.|do_\beta|.L)$ where L is the maximum size of $do_{\alpha \sqcup \beta}$ during the execution of the algorithm. Consequently, the complexity of algorithm 10 becomes $O(|E|) + O(n.m) + O(|F|.|P|.M)$ where E is the set of all event types, M is time complexity of Algorithm 11, n and m are the size of the sequence (i.e., the number of event sets in S) and the number of events per timestamp(i.e., the size of such an event set) and P is the set of frequent episodes of size 1 (single events).

6.1.2.2 An illustrative example

Consider the complex sequence depicted in Fig 6.1 as an illustration, and consider the value of *minsup* to be 0.5. In the initial stage of Algorithm 10, each event type in E is treated as a standalone event episode (i.e., an episode comprising a single event). Subsequently, the algorithm evaluates, for every episode, the set of its occurrences in each event set by scanning the complex event sequence and the expected-support as per definition 17 (refer to lines 2-9). The algorithm subsequently eliminates infrequent single-event episodes based on the support threshold. Table 6.3 lists the set P of frequent episodes obtained by executing lines 1-13.

Table 6.3: Frequent episodes of size 1 for $minsup = 0.5$

Episode	Occurrences	Expected-support
a	$\{[1],[3],[4],[7]\}$	1.9
b	$\{[1],[4],[5]\}$	1.5
c	$\{[2],[4],[6]\}$	2.1
d	$\{[2],[3],[7]\}$	2
e	$\{[3],[6]\}$	1.2

The algorithm develops a replica of the frequent episodes that have already been obtained and then initiates the search for larger episodes using a depth-first approach. Before deciding whether a new episode $\alpha \sqcup \beta$ is frequent, the algorithm determines the probabilistic distinct occurrences of the episode, denoted by $do_{\alpha \sqcup \beta}$. This process is illustrated in the example provided, and the algorithm verifies whether the episode meets the required support. If it does, the episode is incorporated into the set F of frequent episodes, and the search continues (lines 21-25).

To carry out the *probabilistic distinct occurrences recognition* step, Algorithm 10 calls Algorithm 11.

Consider two episodes, $\alpha = a$ and $\beta = b$, with $do_\alpha = \{[1],[3],[4],[7]\}$ and $do_\beta = \{[1],[4],[5]\}$. The first step of each iteration in Algorithm 11 involves creating a new occurrence by combining the timestamps of occurrences of α and β and calculating its probability as the product of their individual probabilities. This is done in the loop from lines 2 to 7.

It is important to note that if the set of distinct occurrences of $do_{\alpha \sqcup \beta}$ is empty, Algorithm 11 proceeds to store the occurrences without performing the loop from lines 10 to 16. As explained previously, the first occurrences $O_\alpha = [1]$ and $O_\beta = [1]$ will result in the occurrence of $\alpha \sqcup \beta$ such that $newocc_{\alpha \sqcup \beta}.timestamps = [1\ 1]$ and $newocc.prob = p(O_\alpha) \times p(O_\beta) = 0.6 \times 0.4 = 0.24$. 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* $newocc_{\alpha \sqcup \beta}.prob = p(O_\alpha) \times p(O_\beta) = 0.2 \times 0.3 = 0.06$ and stores each occurrence in $do_{\alpha \sqcup \beta}$. The occurrence of $\alpha \sqcup \beta$ does not, however, meet the requirements of definition 16 (line 12) in an extreme situation.

As a result, the procedure chooses the instance to overstep in a different way. Thus, the algorithm repeats with the same occurrence of α and chooses the next occurrence of β if the timestamp of the single-event episode β already exists in any previous occurrence of $\alpha \sqcup \beta$. In the absence of this, the algorithm keeps track of the β occurrence and evaluates the combination using the subsequent α occurrence since it intersects with every other instance of $\alpha \sqcup \beta$.

Take two episodes, for example, $\alpha = d$ and $\beta = e$. $do_{\alpha \sqcup \beta} = [2\ 3]$ if the algorithm has the occurrences $O_\alpha = [2]$ and $O_\beta = [3]$. $O_\beta = [6]$ and $O_\alpha = [3]$, however, cannot be combined further since $[2\ 3]$ already contains the timestamp $t = 3$. In order to provide a valid instance of $\alpha \sqcup \beta$ such that $do_{\alpha \sqcup \beta} = \{[2\ 3],[6\ 7]\}$, the algorithm simply keeps the occurrence $O_\beta = [6]$ and advances to the next occurrence, $O_\alpha = [7]$. All frequent episodes are located when the algorithm ends. The final set of frequent episodes for the sequence in Figure 6.1, produced by Algorithm 10, is listed in Table 6.1.

6.1.3 Episode rules extraction

The next step in our method is to identify episode rules from an uncertain sequence. A collection of valid episode rules *R* is the result of the function shown in **Algorithm** 12, which accepts an uncertain sequence *S*, support threshold *minsup*, and confidence threshold *minconf* as inputs.

A broad search space is produced when every potential combination of α and β is considered for building an episode rule. In order to narrow down this search space, the following pruning method was used: Only an episode's super-episodes are potential candidates to be the consequence of valid episode regulations for a particular single-event episode. In the rule generation process, this is obtained by employing the anti-monotony property. Alignment 21.2.

Proposition 21.2. *"Let α and β be two frequent episodes under distinct occurrence-based frequency. If the rule $\alpha \Rightarrow \beta$ is invalid, the episode rule $\alpha \Rightarrow \beta'$ is invalid too for all β' such that $\beta \sqsubseteq \beta'$"*

Proof. Start from a single event episode β such that $\beta \sqsubseteq \beta'$, and it is easy to see that the proof of correctness of that proposition simply depends on the anti-monotony of the distinct occurrence-based support. This makes it easy to demonstrate the confidence of rule $\alpha \Rightarrow \beta'$. Assume $\beta \sqsubseteq \beta'$ and argue against the validity of rule $\alpha \Rightarrow \beta$. $expsup(\beta) \geq expsup(\beta')$ is the anti-monotonicity property derived from Proposition 21.1. $conf(\alpha \Rightarrow \beta) = \frac{|occER(\alpha,\beta)|}{expsup(\alpha)} \geq conf(\alpha \Rightarrow \beta') = \frac{|occER(\alpha,\beta')|}{expsup(\alpha)}$, is the logical conclusion. Nevertheless, because $conf(\alpha \Rightarrow) < minconf$, rule $\alpha \Rightarrow \beta'$ is incorrect owing to the invalidity of $\alpha \Rightarrow \beta$.

A function called **Extract_Episode_Rules_With_Pruning**, which is described in approach 12, implements the suggested approach for finding the episode rules. From the input complex event sequence, this method incorporates a pruning technique based on Proposition 21.2 to provide all the acceptable episode rules.

First, the method initializes the set *R* of valid rules to be determined, the set *P* of frequent single episodes (lines 1-3), and the set of all frequent episodes with regard to the support threshold *minsup*. The method then combines bigger and larger episodes with single event episodes to generate the set of valid consequents with regard to the confidence threshold $minconf$ for each frequent episode α as antecedent of the rule $\alpha \Rightarrow \beta$. The j^{th} single event is used by the algorithm as the first result of an episode rule. Here, if there isn't another potential super-episode of β as a consequence, the episode *root* at line 8 is utilized to retrace the steps. The function $Episode\_Rule\_Support(\alpha, \beta)$, provided by Algorithm 13, is called by the algorithm to determine the episode rule support for each episode β. It also computes the rule's confidence based on definition 21: The method inserts the episode $\alpha \Rightarrow \beta$ into *R* and updates the *root* episode for the later combination of the consequents (lines 11–19) if the confidence exceeds or is equal

to the confidence threshold. If there are no more candidates for single event episodes, the algorithm ends; if not, it uses the subsequent j^{th} episode from P to update β and make it larger (lines 20–23).

From the input complex event sequence, it is evident that Algorithm 12 creates all acceptable episode rules because it systematically searches the rule space and only ignores rules that violate Proposition 21.2. In other words, only unsound rules are disregarded.

6.1.3.1 Time complexity

The extraction of episode rules with pruning is the final step of the proposed approach, consisting of UEMDO and UEMDO-P. The overall approach consists of first calling Algorithm 10 to find all the frequent episodes with time complexity denoted as N. Then, the algorithm 13 is called to generate all valid rules with respect to a confidence threshold $minconf$. This search continue as long as there exists a valid combination that respect definition 11 with time complexity $O(|do_\alpha|.|do_\beta|^2)$. Otherwise, the algorithm stops when it finds episodes that do not meet that definition. Based on this observation and the previous discussion of the complexity of Algorithm 10 with respect to the complexity of Algorithm 13 expected-support, the overall time complexity is: $O(|F|.|P|^2.|do_\alpha|.|do_\beta|^2)+N)$ where F is the set of all frequent episodes under distinct occurrences and N the time complexity of Algorithm 10 called in line 1 (see Algorithm 12).

6.1.3.2 An illustrative example

An example is provided to show how to use Algorithm 12 to find the set of all episode rules in the sequence S showed in Fig. 6.1 for $minsup = 0.5$ and $minconf = 0.1$. As previously said, the algorithm first determines the collection of frequent episodes (see Table 6.1). Next, it takes the sequence S and builds the set P of frequent episodes of size 1 (line 3), that is, $P = \{a,b,c,d,e\}$. Subsequently, the system will look for episode rules. In the case of episode $\alpha = a$, the algorithm identifies all rules in which the antecedent is $\alpha = a$ ($\alpha \in F$). It then treats $\beta = b$ ($\beta \in P$) as the first consequent and the root of potential consequent (line 4–10), where $do_\alpha = \{[1],[3],[4],[7]\}\{[1],[4],[5]\}$ is the value of do_β. Next, using the definitions of 19 and 20, the algorithm determines the expected-support of the episode rule $ER = \alpha \rightarrow \beta$ (line 13). The expected-support is computed in this way: for each occurrence $O_i \in do_\alpha$, the occurrence $O_j \in_\beta$

is found by applying Algorithm 13, where $do_\beta = \{[1],[4],[5]\}$ and $do_\alpha = \{[1],[3],[4],[7]\}$ such that $end(O_i) < end(O_j)$ and $start(O_i) < start(O_j)$. The first occurrence of the episode rule $ER = \alpha \rightarrow \beta$ is therefore [1 4]; it cannot be [1 1] because $O_\alpha = [1]$ and $O_\beta = [1]$ do not satisfy prior requirements. Consequently, the algorithm advances to β ($O_\beta = [4]$), which occurs twice and results in the occurrence of an episode rule that is valid. The algorithm then advances to the subsequent instances of $O_\beta = [5]$ and $O_\alpha = [3]$ of β and α, respectively, resulting in vector [3 5] being a legitimate instance of the episode rule.

For each given rule $\alpha \rightarrow \beta$, the set of occurrences is $ER - occ(\alpha \rightarrow \beta) = \{[1\ 4],[3\ 5]\}$. Ultimately, for each $O_k \in ER - occ(\alpha \rightarrow \beta)$, the expected-support is computed as follows: $expsup(\alpha \rightarrow \beta) = p(O_1) + p(O_2) = 0.34$. Subsequently, Algorithm 12 proceeds with its computation by determining the confidence on line 14. The current consequent β becomes the root of probable consequents if the confidence is higher than the $minconf$ threshold. In this case, the procedure is repeated and the current rule is deemed legitimate. Next, it is connected with the next single event episode $\gamma = c$ from the set P. Observe that the root replaces the legitimate consequent to join if the consequent is not common.

The rule $a \rightarrow \beta$ is valid for $minconf = 0.1$, and since $\beta = b$ is the root for subsequent consequents, it is linked with episode $\gamma = c$. Thus, $a \rightarrow bc$ is the next episode rule to be examined. According to the assertion 21.2, $\beta = b$ will be connected with $\delta = d$ if the previous rule is invalid with regard to $minconf$.

The full list of valid episode rules that the algorithm found is displayed in Table 6.2.

6.2 Experimental study

In order to show the efficiency of our approach, several experiments have been executed on both synthetic and real datasets [114]. The results compares only the performances of our approach since there exist no prior work of frequent episodes mining from uncertain complex event sequences. 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.

Algorithm 12: Extracting Episode rules with pruning

Input: S: complex event sequence on E (event types set), $minsup$: support threshold, $minconf$: confidence threshold

Output: R: complete set of episode rules.

```
F ← FrequentEpisodes(minsup)
R ← ∅
P ← {γ | γ ∈ F ∧ ||γ|| = 1}
for each α in F do
    j ← 0
    while j < |P| do
        β ← P[j]
        root ← β
        k ← j
        stop ← false
        while not stop do
            if β ∈ F then
                ruleExpectedSupport ← EpisodeRuleExpectedSupport(α, β)
                conf ← ruleExpectedSupport / α.sup
                if conf ≥ minconf then
                    R ← R ∪ {α → β}
                    root ← β
                else
                    β ← root
                end
            else
                β ← root
            end
            if k > |P| then
                stop ← true
            else
                β ← β ⊔ P[k]
                k ← k + 1
            end
        end
        j ← j + 1
    end
end
return R
```

Algorithm 13: Episode Rule Expected-Support

Input: Two Episodes α and β
Output: *ruleExpectedSupport*: The expected support of the rule $\alpha \Rightarrow \beta$.

```
{% Initialization %}
ruleSupport ← 0
i ← 1, j ← 1
while i ≤ |do_α| do
    stop ← false
    while j ≤ |do_β| and not stop do
        for each O_k in do_β s.t: j < k ≤ |do_β| do
            if start(O_i) < start(O_j) and end(O_i) < end(O_j) then
                i ← i + 1
                ruleExpectedSupport ← ruleExpectedSupport + (O_i.prob × O_j.prob)
                stop ← true
            end
        end
        j ← k
    end
end
return ruleExpectedSupport
```

6.2.1 Uncertain datasets used for the test

As mentioned before, we used many synthetic uncertain datasets. Three primary factors are used to produce these datasets at random:

- The number of events and the duration of the complex sequence,
- the event types count and
- the event sets maximal size in the sequence.

The real datasets were acquired by extending pre-existing transaction databases into intricate event sequences from the SPMF datasets collection [115]. To be more specific, every item is seen as an event, every transaction (itemset) as an event set, and every transaction database as a complicated series. Additionally, a number is assigned to each event set in order to represent its timestamp.

The first dataset, named *"OnlineRetail"*, has 2,603 different event kinds and 541,880 event sets. The fourth dataset, titled *"Mushrooms"*, has 8,416 event sets and 119 event types; the fifth and final dataset, titled *"FruitHut"*, has 181970 event sets and 9390 event types. The second

dataset, *"Retail"s*, is a complex sequence with 88,162 event sets and 16,470 event types. A probability ranging from 0 to 1 was assigned to every event. The datasets were chosen because they offer a variety of features and are well-liked benchmark datasets for assessing pattern mining methods. These datasets together with the synthetic datasets can provide a comprehensive picture of the algorithms' performance under various conditions.

6.2.2 Discussion of results

The two processes that are being evaluated in this part are the mining of frequent episodes (part 6.2.2.1) and the generation of valid episode rules (Section 6.2.2.2).

When it comes to frequent episode mining, the performance analysis takes into account the variance based on the support threshold of both synthetic and actual datasets of:

- The memory consumed throughout the procedure,
- the number of frequent episodes,
- the size of the biggest frequent episode, and
- the run-time (in seconds).

For episode rules mining, we compare, for both synthetic and real datasets, between the baseline algorithm (UEMDO) and that using the pruning strategy (UEMDO-P) proposed in this paper, in terms of runtime and memory cost.

6.2.2.1 Frequent episodes mining

Figures 6.2 and 6.3 show the results in terms of runtime and memory usage in megabits of the frequent episode generation step described by Algorithm 10 of UEMDO for different values of the support threshold *minsup* for both synthetic and real datasets. It is easy to see that the runtime and memory usage are sensitive to different values of *minsup* where they decrease when the support threshold increases for the three sequences. This phenomenon is caused by the strong application of the anti-monotony of the distinct occurrence-based support property because the support of super-episodes is at most the support of one of its sub-episodes; hence, when *minsup* value increases, the algorithm consumes less time and memory to proceed with larger episodes.

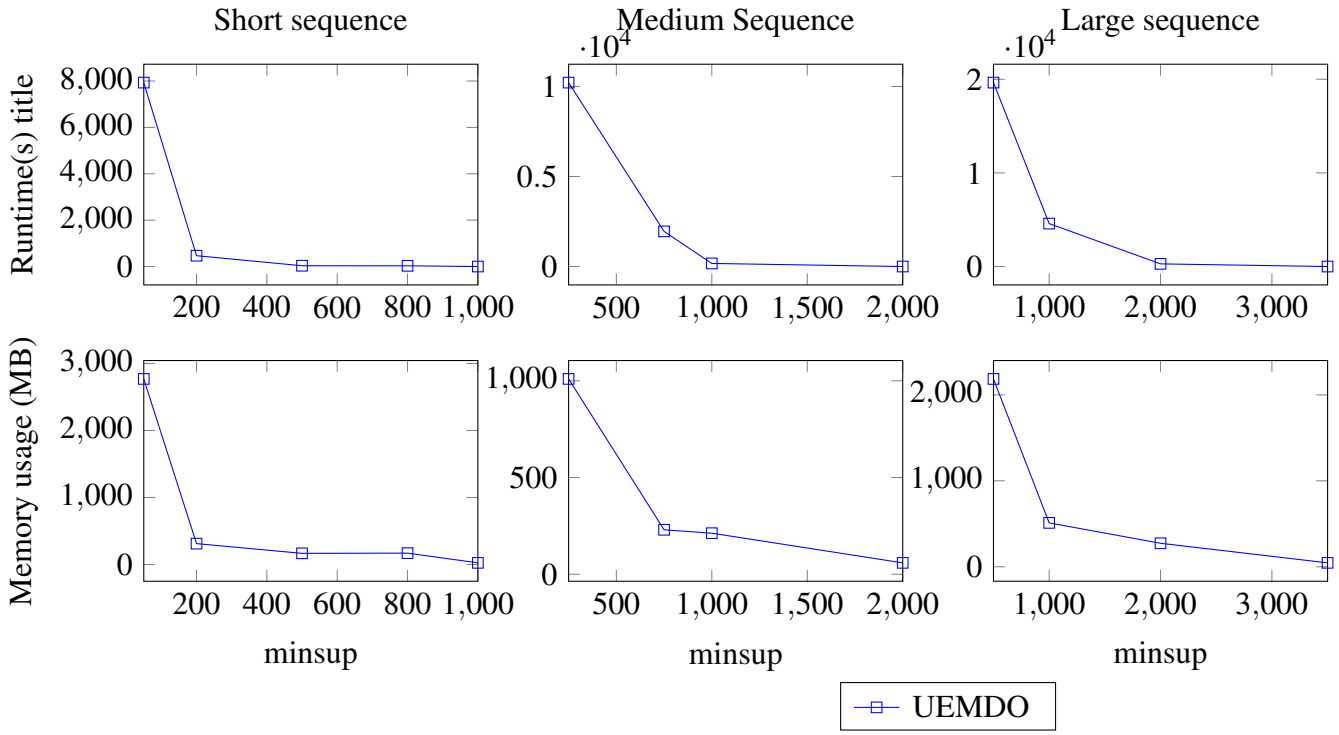


Figure 6.2: Influence of *minsup* on execution time and memory usage for frequent episode generation on synthetic data sets

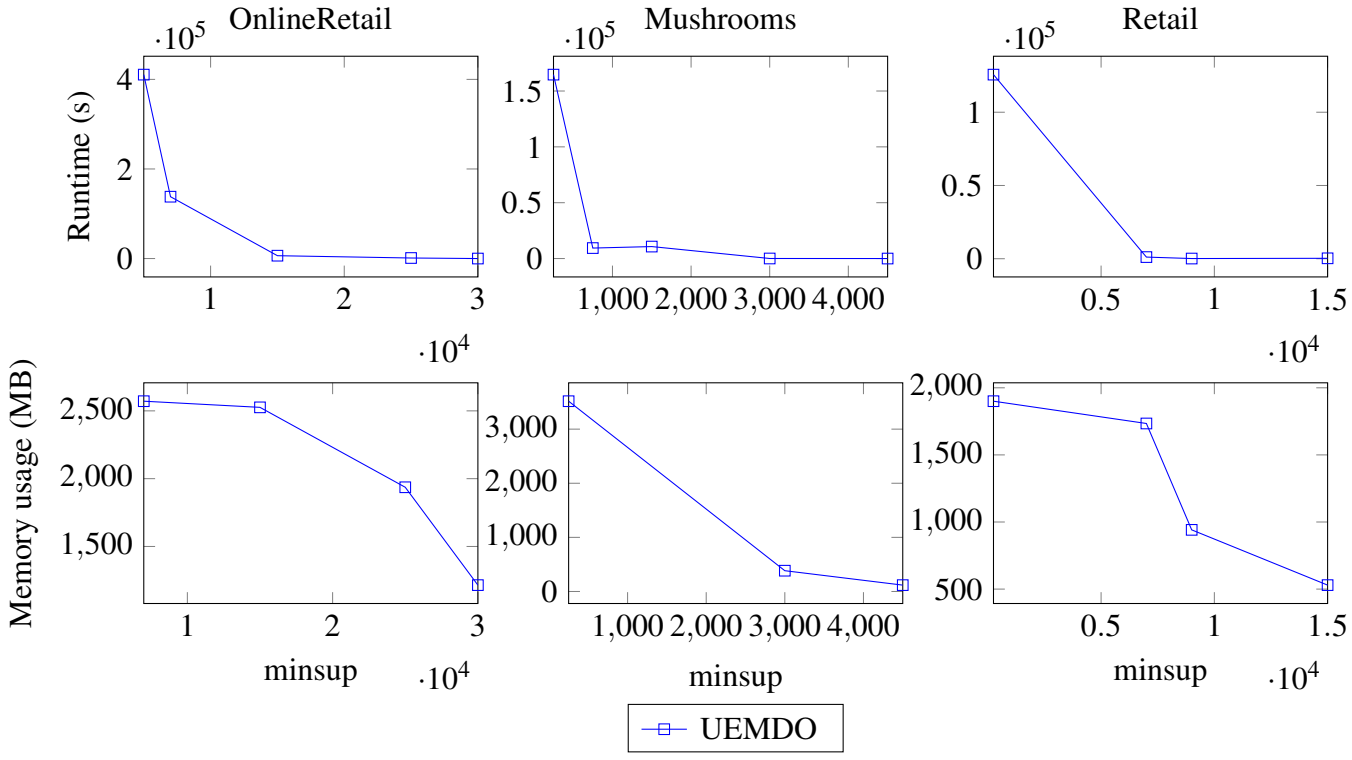


Figure 6.3: Influence of *minsup* on the execution time and memory usage for frequent episode generation on real data sets

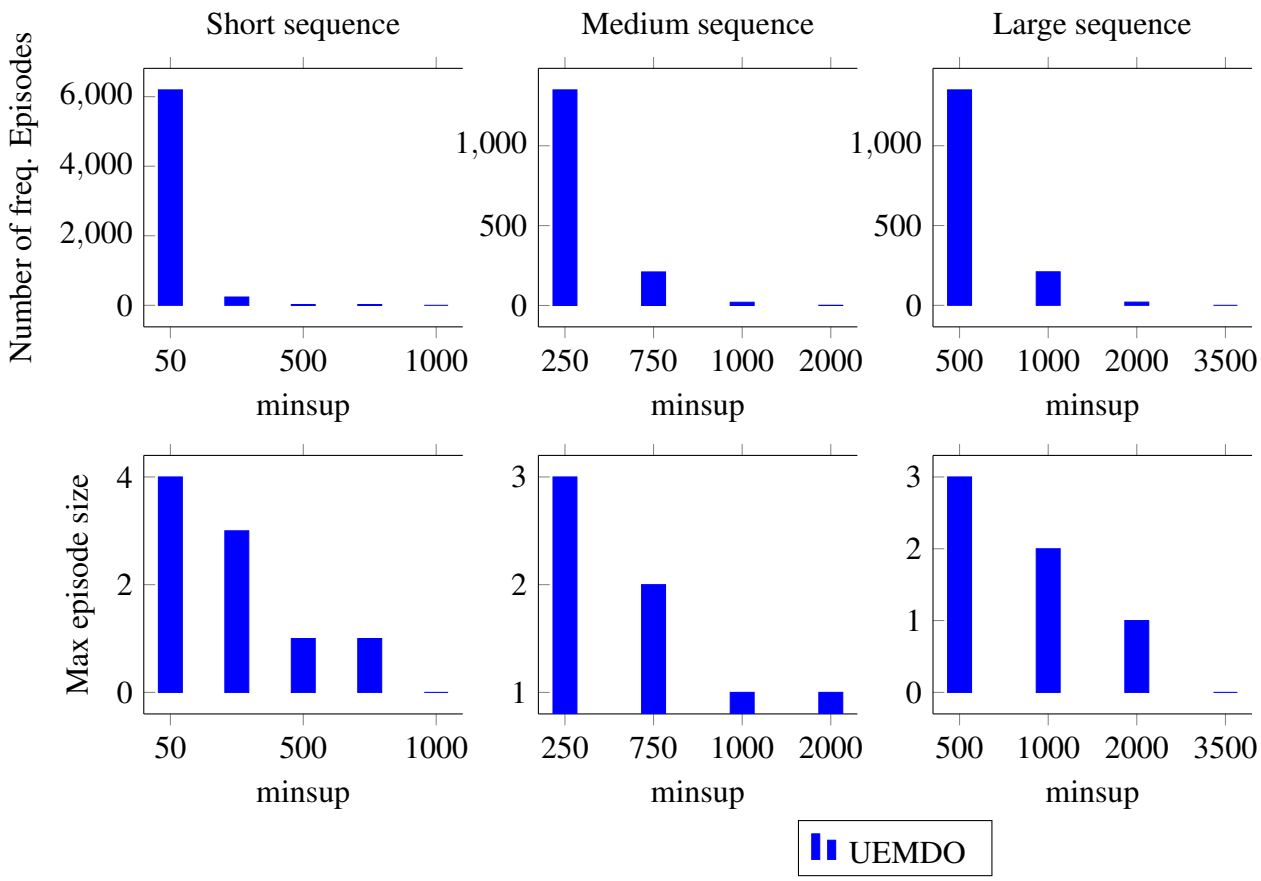


Figure 6.4: Influence of *minsup* on the number of generated frequent episodes and the size of the largest frequent episode on synthetic datasets

The figures depicted in 6.4 and 6.5 offer a comparative analysis of the number of frequent episodes and the size of the largest frequent episode generated by the frequent episode generation step described in Algorithm 10, on both synthetic and real datasets, respectively. These graphs exhibit an inverse proportion between the two values. This observation is consistent with the findings related to runtime and memory usage, indicating that the anti-monotony property performs exceptionally well in our algorithm. This phenomenon suggests that for lower support values, UEMDO generates episodes with larger sizes, thereby incorporating more information and relationships between events in the sequence, despite the data's uncertain nature.

6.2.2.2 Generation of episode rules

The next step of our approach, after generating the frequent episodes set, is the generation of episode rules relating pairs of frequent episodes with respect to the minimum confidence threshold. Because of the absence of other techniques for discovering parallel episodes and/or generating episode rules from complex uncertain sequences, we focus our evaluation on the

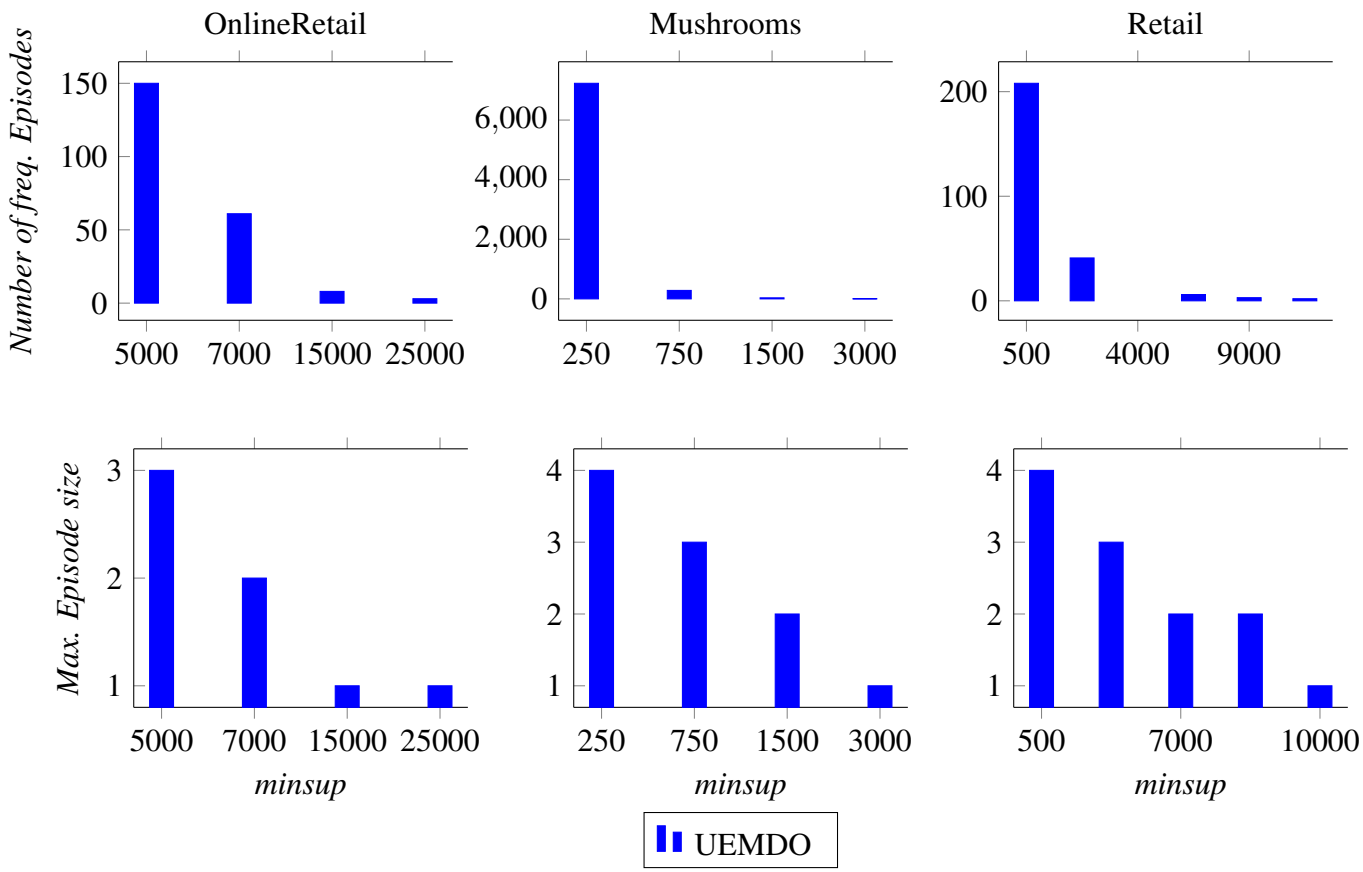


Figure 6.5: Influence of *minsup* on the number of generated frequent episodes and the size of the largest frequent episode on real datasets

comparison between our baseline algorithm (UEMDO) and the modified version (UEMDO-P) described in Algorithm 12 which performs episode rule search by incorporating a pruning strategy based on Property 21.2. Note that these two versions generate the same set of episode rules, and hence, the same number of episode rules.

Figures 6.6 and 6.7 show the variations in runtime and memory usage of the two variants, UEMDO and UEMDO-P, of our algorithm according to different confidence threshold values on both real and synthetic datasets. The obtained results confirm the effectiveness of our pruning strategy in reducing runtime considerably. Our pruning strategy enables the algorithm to discontinue the search for a particular antecedent when it locates the most significant invalid episode rules associated with it. Subsequently, it backtracks with other consequents until it processes all frequent episode sets. Contriraly, the baseline approach examines every element in the sequence for each potential antecedent, leading to an extended search process that consumes excessive time to complete.

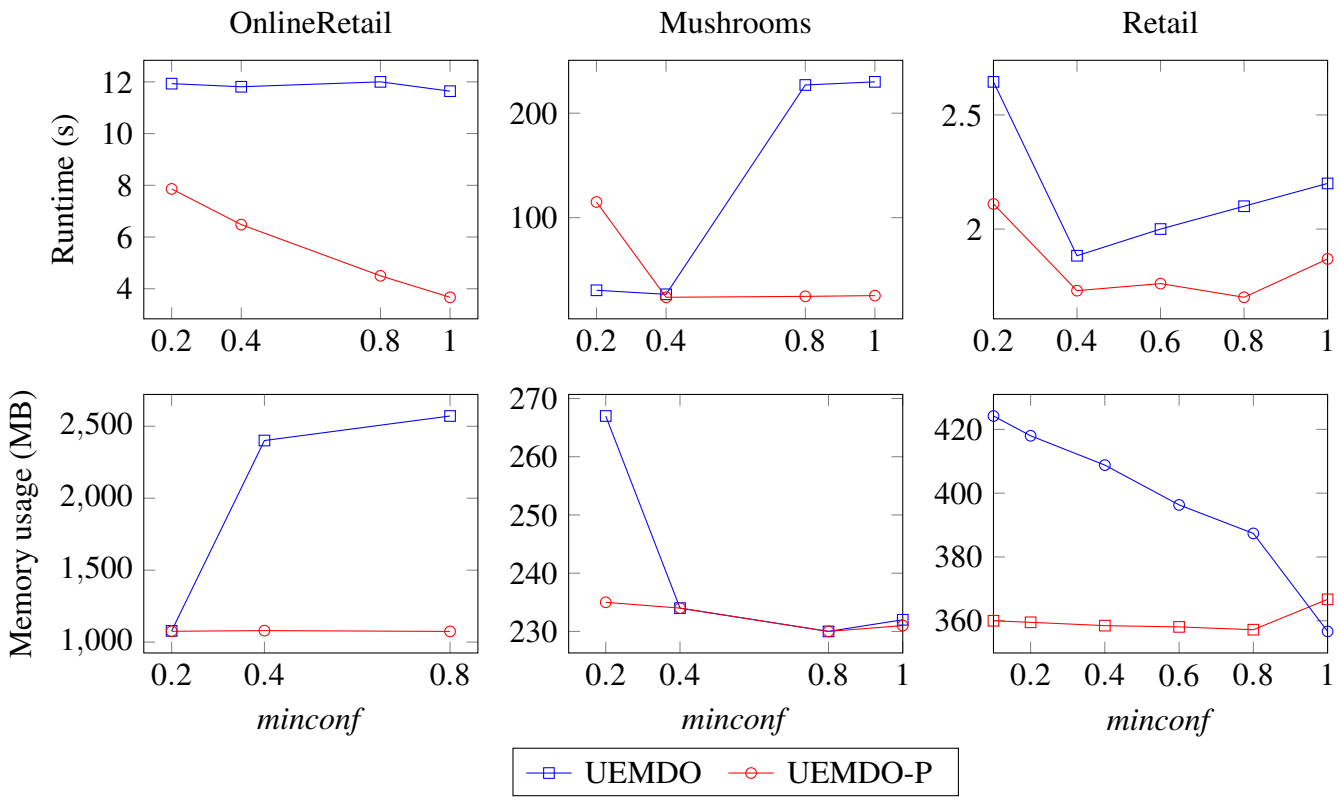


Figure 6.6: Influence of *minconf* on (a) execution time (b) number of frequent episodes and (c) memory usage on real datasets

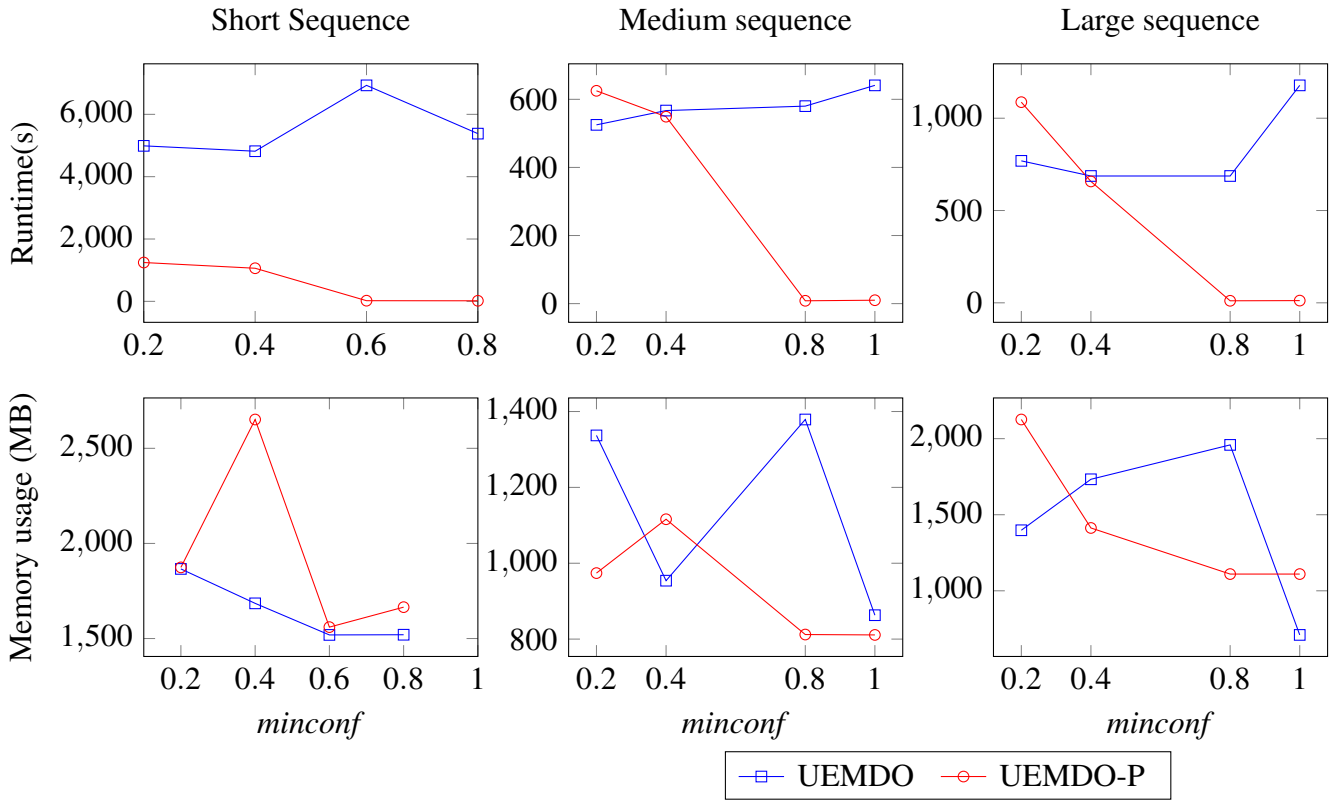


Figure 6.7: Influence of *minconf* on execution time and memory usage of episode rules generation on synthetic datasets

6.2.2.3 Discussion of some discovered patterns

Using the UEMDO-P algorithm, several interesting rules were discovered in real data. For instance, some rules found in the *FruitHut* dataset are shown in Table 6.4. These rules represent temporal relationships between items that ha ve been purchased by customers in transactions from a fruit store.

Table 6.4: Example rules from $FruitHut$

$Rule$	$conf(\alpha \rightarrow \beta)$
Tomatoes Truss $\rightarrow$ Crimson Grapes	41.46%
Beans green $\rightarrow$ Onion spring	50.14%
Mushroom cup $\rightarrow$ Lettuce Iceberg, Beans green	21.23%
Onion Brown, Lettuce Iceberg $\rightarrow$ Eggplant	50.66%
Lettuce Iceberg, Beans green $\rightarrow$ Broccoli	50.52%
Potato Brushed, Lettuce Iceberg $\rightarrow$ Potato sweet gold	50.89%

These episode rules are interesting as they show that some items are often bought in a specific order, which can reveal customer preferences, habits or needs. For example, the rule Tomatoes Truss $\rightarrow$ Crimson Grapes indicates that customers who buy tomatoes truss are likely to buy crimson grapes within a given time interval afterwards, with a confidence of 41.46% that is relatively high. A reason could be that customers are preparing salads or sandwiches that require both ingredients, or that they enjoy the taste of both products. Similarly, another rule, Beans green $\rightarrow$ Onion spring, indicates that customers buying onion spring are likely to buy beans green afterward. Generally, discovering such episode rules provide insights on customer behaviro and with further analysis could lead to develop improved marketing strategies such as using cross-selling, recommendation, or promotion.

6.3 Conclusion

In this chapter, we addressed the problem of mining frequent parallel episodes from complex uncertain sequences of events using the expected-support approach under distinct occurrence-based frequency. We have presented an efficient depth-first strategy called UEMDO. This algorithm was then extended to also mine episode rules. The results show the effectiveness of our novel algorithm in reducing runtime and memory cost. Our novel algorithm can be used, for instance, in recommendation systems for predicting the future actions of some attacks in

communication networks.

Chapter 7

General Conclusion

Data-mining is a large field that aims to discover interesting insights from a large volume of data. This thesis brings new contributions to an important sub-field of data mining that aims at extracting important patterns from data. We have focused particularly on extracting frequent episodes from large sequences of events.

Frequent episode mining framework was designed to analyze a large sequence of events. It is an active area of research in recent decades, as evidenced by numerous published studies. It helps to analyze temporal data, understand the behavior of systems, detect abnormalities, and predict the future. In this thesis, we have presented an up-to-date state-of-the-art of this framework and its variants, as well as novel approaches for certain and uncertain data that are summarized in what follows:

- We proposed a novel approach for episode and episode rule mining from certain sequences of events. It includes two algorithms: The first one mines frequent serial episodes under non-overlapping occurrence based frequency from an event sequence. The second algorithm uses the first one to mine episode rules based on a depth-first search strategy. The algorithm yielded good results compared to existing methods. The previous contribution is applicable for prediction tasks in several domains, such as medicine and stock market buy/sell actions, because of its rules form that can describe the prefix extensibility of sub-sequences.

- Because multiple frequency definitions exist in episode mining, we have proposed novel

algorithms for frequent episode and episode rule discovery under the distinct occurrence-based frequency called EMDO and EMDO-P, respectively. This frequency definition may detect episodes that may overlap without sharing common timestamps. This approach captures episodes when the order of a given episode is not considered. The proposed rules are very useful for analyzing the relationships between a wide range of pairs of episodes and include the rules proposed in the first contribution. For example, it can be applied to analyze the click logs of any website to obtain a set of rules that help users recommend the next pages to visit in order to enhance the performance of such a website.

- To address data uncertainty, some proposed methods calculate the existential probability of episodes in an event sequence. However, few studies deal with event probabilities. As a novel contribution, we proposed an extension of EMDO and EMDO-P to existential probability episodes in sequence with uncertain events. The new extension calculates the expected support for an episode based on the probability of its distinct occurrences.

Research opportunities

There are several opportunities for research on episode mining. Some of the studies mentioned in this thesis include:

- *Enhancing the performance of existing episode mining algorithms*, Many algorithms are still resource-intensive, especially when dealing with long or intricate event sequences. Furthermore, most algorithms developed for episode mining have only been applied in centralized systems. Therefore, creating algorithms that operate effectively on distributed systems or developing parallel mining algorithms to improve speed and scalability is a significant challenge in the field of episode mining.

- *Extending the existing algorithms to consider more complex types of episodes and episode rules*, Many existing algorithms primarily focus on identifying a single type of episodes and/or episode rules. However, numerous applications involve numerous simultaneous events. Recent research has begun to address this issue, such as in medical applications [59]. Nevertheless, this area of study requires further investigation. Furthermore, there exist a few work that consider episodes with partial order, which really correspond to the reality of many systems.

- *Finding more applications that depend on episode mining approach*, Most current research focuses on developing algorithms based on theoretical concepts rather than practical application. As a result, the process of analyzing data using temporal analysis with episodes and episode rule discovery is an understudied topic. To address this gap, it is recommended to use episode mining to analyze any recorded data, particularly in fields such as cybersecurity, medicine, economics, and finance, where the order of elements is critical.

- Another very important future research area is the find extensions of existing state-of-the-art methods to incorporate techniques of deep learning which is is today a powerful and trending field in Artificial Intelligence.

References

[1] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *WIREs Data Mining and Knowledge Discovery*, 2017.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB 1994*, Santiago de Chile, Chile, 12-15 September, 1994, p. 487–499.

[3] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. on Knowl. and Data Eng.*, vol. 12, no. 3, p. 372–390, may 2000. [Online]. Available: https://doi.org/10.1109/69.846291

[4] P. Fournier Viger, C.-W. Lin, U. Rage, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, pp. 54–77, 02 2017.

[5] T.-T. Pham, J. Luo, T.-P. Hong, and B. Vo, "An efficient method for mining non-redundant sequential rules using attributed prefix-trees," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 88–99, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197614000554

[6] P. Fournier-Viger, C.-W. Wu, V. S. Tseng, L. Cao, and R. Nkambou, "Mining partially-ordered sequential rules common to multiple sequences," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2203–2216, 2015.

[7] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," in *Advances in Knowledge Discovery and Data Mining*, V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen, and H.-Y. Kao, Eds. Cham: Springer International Publishing, 2014, pp. 40–52.

[8] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Advances in Database Technology — EDBT '96*, P. Apers, M. Bouzeghoub, and G. Gardarin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 1–17.

[9] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: the prefixspan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.

[10] P. Fournier-Viger, C.-W. Wu, V. S. Tseng, L. Cao, and R. Nkambou, "Mining partially-ordered sequential rules common to multiple sequences," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 8, pp. 2203–2216, 2015.

[11] P. Fournier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, "Erminer: Sequential rule mining using equivalence classes," in *Advances in Intelligent Data Analysis XIII*, H. Blockeel, M. van Leeuwen, and V. Vinciotti, Eds. Cham: Springer International Publishing, 2014, pp. 108–119.

[12] H. Mannila, H. Toivonen, and A. Verkamo, "Discovery of frequent episodes in event sequences," *Data Min. Knowl. Discov.*, vol. 1, pp. 259–289, 07 1997. [Online]. Available: https://doi.org/10.1023/A:1009748302351

[13] A. Avinash, L. Srivatsan, and P. S. Sastry, "A unified view of the apriori-based algorithms for frequent episode discovery," *Knowledge and Information Systems*, vol. 31, pp. 223–350, 2012. [Online]. Available: https://doi.org/10.1007/s10115-011-0408-2

[14] A. Avinash, A. Ibrahim, and P. Sastry, "Pattern-growth based frequent serial episode discovery," *Data and Knowledge Engineering*, vol. 87, pp. 91–108, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X13000724

[15] G. Liao, X. Yang, S. Xie, P. S. Yu, and C. Wan, "Two-phase mining for frequent closed episodes," in *Web-Age Information Management*, B. Cui, N. Zhang, J. Xu, X. Lian, and D. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 55–66.

[16] W. Zhou, H. Liu, and H. Cheng, "Mining closed episodes from event sequences efficiently," in *Advances in Knowledge Discovery and Data Mining*, M. J. Zaki, J. X. Yu,

B. Ravindran, and V. Pudi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 310–318.

[17] H. Zhu, P. Wang, W. Wang, and B. Shi, "Discovering frequent closed episodes from an event sequence," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[18] G. Aliberti, A. Colantonio, R. Di Pietro, and R. Mariani, "Expedite: Express closed itemset enumeration," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3933–3944, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414008100

[19] Y. Wu, C. Zhu, Y. Li, L. Guo, and X. Wu, "Netncsp: Nonoverlapping closed sequential pattern mining," *Knowledge-Based Systems*, vol. 196, p. 105812, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120301945

[20] L. Wen, "An efficient algorithm for mining frequent closed itemset," in *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No.04EX788)*, vol. 5, 2004, pp. 4296–4299 Vol.5.

[21] E.-Z. Guan, X.-Y. Chang, Z. Wang, and C.-G. Zhou, "Mining maximal sequential patterns," in *2005 International Conference on Neural Networks and Brain*, vol. 1, 2005, pp. 525–528.

[22] X. Ao, H. Shi, J. Wang, L. Zuo, H. Li, and Q. He, "Large-scale frequent episode mining from complex event sequences with hierarchies," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 4, jul 2019. [Online]. Available: https://doi.org/10.1145/3326163

[23] C. Gao, J. Wang, Y. He, and L. Zhou, "Efficient mining of frequent sequence generators," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1051–1052. [Online]. Available: https://doi.org/10.1145/1367497.1367651

[24] L. Szathmary, P. Valtchev, A. Napoli, R. Godin, A. Boc, and V. Makarenkov, "A fast compound algorithm for mining generators, closed itemsets, and computing links between equivalence classes," *Annals of Mathematics and Artificial Intelligence*, vol. 70,

02 2014.

[25] P. Fournier-Viger, J. Chun-Wei Lin, T. Truong-Chi, and R. Nkambou, *A Survey of High Utility Itemset Mining*. Cham: Springer International Publishing, 2019, pp. 1–45. [Online]. Available: https://doi.org/10.1007/978-3-030-04921-8_1

[26] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

[27] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7419–7424, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417410014454

[28] J.-F. Qu, M. Liu, and P. Fournier-Viger, *Efficient Algorithms for High Utility Itemset Mining Without Candidate Generation*. Cham: Springer International Publishing, 2019, pp. 131–160. [Online]. Available: https://doi.org/10.1007/978-3-030-04921-8_5

[29] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "A novel approach for mining high-utility sequential patterns in sequence databases," *ETRI Journal*, vol. 32, no. 5, pp. 676–686, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.4218/etrij.10.1510.0066

[30] B.-E. Shie, J.-H. Cheng, K.-T. Chuang, and V. S. Tseng, "A one-phase method for mining high utility mobile sequential patterns in mobile commerce environments," in *Advanced Research in Applied Artificial Intelligence*, H. Jiang, W. Ding, M. Ali, and X. Wu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 616–626.

[31] T. Dinh, B. Le, P. Fournier Viger, and V.-N. Huynh, "An efficient algorithm for mining periodic high-utility sequential patterns," *Applied Intelligence*, vol. 48, p. 4694–4714, 12 2018.

[32] G.-C. Lan, T.-P. Hong, V. S. Tseng, and S.-L. Wang, "Applying the maximum utility measure in high utility sequential pattern mining," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5071–5081, 2014. [Online]. Available: https:

//www.sciencedirect.com/science/article/pii/S0957417414000852

[33] C.-W. Wu, Y.-F. Lin, P. S. Yu, and V. S. Tseng, "Mining high utility episodes in complex event sequences," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 536–544. [Online]. Available: https://doi.org/10.1145/2487575.2487654

[34] G. Guo, L. Zhang, Q. Liu, E. Chen, F. Zhu, and C. Guan, "High utility episode mining made practical and fast," in *Advanced Data Mining and Applications*, X. Luo, J. X. Yu, and Z. Li, Eds. Cham: Springer International Publishing, 2014, pp. 71–84.

[35] P. Fournier-Viger, P. Yang, J. C.-W. Lin, and U. Yun, *HUE-Span: Fast High Utility Episode Mining*. Cham: Springer International Publishing, 2019, pp. 169–184.

[36] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, and Y.-K. Lee, "Discovering periodic-frequent patterns in transactional databases," in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 242–253.

[37] R. Uday Kiran and P. Krishna Reddy, "Towards efficient mining of periodic-frequent patterns in transactional databases," in *Database and Expert Systems Applications*, P. G. Bringas, A. Hameurlain, and G. Quirchmayr, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 194–208.

[38] P. Fournier-Viger, Z. Li, J. C.-W. Lin, R. U. Kiran, and H. Fujita, "Discovering periodic patterns common to multiple sequences," in *Big Data Analytics and Knowledge Discovery*, C. Ordonez and L. Bellatreche, Eds. Cham: Springer International Publishing, 2018, pp. 231–246.

[39] P. Fournier-Viger, P. Yang, Z. Li, J. C.-W. Lin, and R. U. Kiran, "Discovering rare correlated periodic patterns in multiple sequences," *Data & Knowledge Engineering*, vol. 126, p. 101733, 2020, special Issue DAWAK 2018 (Data Warehousing and Knowledge Discovery). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X19304343

[40] J. Soulas and P. Lenca, "Periodic episode discovery over event streams," in *Progress in Artificial Intelligence*, F. Pereira, P. Machado, E. Costa, and A. Cardoso, Eds. Cham: Springer International Publishing, 2015, pp. 547–559.

[41] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, "A fast algorithm for finding frequent episodes in event streams," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 410–419. [Online]. Available: https://doi.org/10.1145/1281192.1281238

[42] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua, and Z. Min, *A Survey of Machine Learning for Network Fault Management*. Cham: Springer International Publishing, 2021, pp. 1–27. [Online]. Available: https://doi.org/10.1007/978-3-030-66288-2_1

[43] O. Ouarem, F. Nouioua, and P. Fournier-Viger, "Mining episode rules from event sequences under non-overlapping frequency," in *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices*, H. Fujita, A. Selamat, J. C.-W. Lin, and M. Ali, Eds. Cham: Springer International Publishing, 2021, pp. 73–85.

[44] M.-Y. Su, "Discovery and prevention of attack episodes by frequent episodes mining and finite state machines," *J. Netw. Comput. Appl.*, vol. 33, no. 2, p. 156–167, mar 2010. [Online]. Available: https://doi.org/10.1016/j.jnca.2009.10.003

[45] A. Ng and A. W.-c. Fu, "Mining frequent episodes for relating financial events and stock trends," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 27–39.

[46] M.-F. Wang, Y.-C. Wu, and M.-F. Tsai, "Exploiting frequent episodes in weighted suffix tree to improve intrusion detection system," in 22^{nd} *International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, 2008, pp. 1246–1252.

[47] M.-Y. Su, "Internet worms identification through serial episodes mining," in *ECTICON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2010, pp. 132–136.

[48] S. Laxman, B. Shadid, P. Sastry, and K. Unnikrishnan, "Temporal data mining for root-cause analysis of machine faults in automotive assembly lines," *CoRR*, 05 2009. [Online]. Available: https://arxiv.org/abs/0904.4608

[49] M. Qin and K. Hwang, "Frequent episode rules for internet anomaly detection," in *Third IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings.*, 2004, pp. 161–168.

[50] H. Mannila and H. Toivonen, "Discovering generalized episodes using minimal occurrences," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 146–151.

[51] A. Achar, S. Laxman, R. Viswanathan, and P. Sastry, "Discovering injective episodes with general partial orders," *Data Mining and Knowledge Discovery*, vol. 25, pp. 67–108, 07 2012.

[52] X. Ma, H. Pang, and K.-L. Tan, "Finding constrained frequent episodes using minimal occurrences," in *Fourth IEEE International Conference on Data Mining (ICDM'04)*, 2004, pp. 471–474.

[53] K. Amphawan, J. Soulas, and P. Lenca, "Mining top-k regular episodes from sensor streams," *Procedia Computer Science*, vol. 69, pp. 76–85, 2015, the 7th International Conference on Advances in Information Technology. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915031725

[54] A. Achar and P. S. Sastry, "Discovering frequent chain episodes," *Knowledge and Information Systems*, pp. 447–494, 2019. [Online]. Available: https://doi.org/10.1007/s10115-019-01349-y

[55] H. Zhu, P. Wang, X. He, Y. Li, W. Wang, and B. Shi, "Efficient episode mining with minimal and non-overlapping occurrences," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 1211–1216.

[56] R. Gwadera, M. Atallah, and W. Szpankowski, "Reliable detection of episodes in event sequences," in *Third IEEE International Conference on Data Mining*, 2003, pp. 67–74.

[57] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for

intrusion detection," *Int. J. Intell. Syst.*, vol. 15, pp. 687–703, 08 2000.

[58] H. K. Dai and Z. Wang, "Mining serial-episode rules using minimal occurrences with gap constraint," in *Computational Science and Its Applications – ICCSA 2013*, B. Murgante, S. Misra, M. Carlini, C. M. Torre, H.-Q. Nguyen, D. Taniar, B. O. Apduhan, and O. Gervasi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 562–572.

[59] Y. Cao, D. Patnaik, S. Ponce, J. Archuleta, P. Butler, W. Feng, and N. Ramakrishnan, "Parallel mining of neuronal spike streams on graphics processing units," *International Journal of Parallel Programming*, vol. 40, pp. 1–28, 12 2011.

[60] K.-Y. Huang and C.-H. Chang, "Efficient mining of frequent episodes from complex sequences," *Information Systems*, vol. 33, no. 1, pp. 96–114, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437907000506

[61] K. Iwanuma, R. Ishihara, Y. Takano, and H. Nabeshima, "Extracting frequent subsequences from a single long data sequence a novel anti-monotonic measure and a simple on-line algorithm," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005, pp. 8 pp.–.

[62] N. Méger and C. Rigotti, "Constraint-based mining of episode rules and optimal window sizes"," in *Knowledge Discovery in Databases: PKDD 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 313–324.

[63] X. Ao, P. Luo, J. Wang, F. Zhuang, and Q. He, "Mining precise-positioning episode rules from event sequences," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, 2017, pp. 83–86.

[64] Y.-F. Lin, C.-W. Wu, C.-F. Huang, and V. S. Tseng, "Discovering utility-based episode rules in complex event sequences," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5303–5314, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415001219

[65] 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 *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2018,

pp. 890–899.

[66] T. Katoh, H. Arimura, and K. Hirata, "An efficient depth-first search algorithm for extracting frequent diamond episodes from event sequences," *IPSJ Online Transactions*, vol. 3, pp. 1–12, 01 2010.

[67] P. Fournier-Viger, Y. Chen, F. Nouioua, and J. C.-W. Lin, "Mining partially-ordered episode rules in an event sequence," in *Intelligent Information and Database Systems*, N. T. Nguyen, S. Chittayasothorn, D. Niyato, and B. Trawiński, Eds. Cham: Springer International Publishing, 2021, pp. 3–15.

[68] Y. Chen, P. Fournier-Viger, F. Nouioua, and Y. Wu, "Mining partially-ordered episode rules with the head support," in *Big Data Analytics and Knowledge Discovery*, M. Golfarelli, R. Wrembel, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer International Publishing, 2021, pp. 266–271.

[69] O. Ouarem, F. Nouioua, and P. Fournier-Viger, "A survey of episode mining," *WIREs Data Mining and Knowledge Discovery*, vol. 14, no. 2, p. e1524, 2024. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1524

[70] C. Borgelt and D. Picado-Muiño, "Finding frequent patterns in parallel point processes," in *Advances in Intelligent Data Analysis XII*, A. Tucker, F. Höppner, A. Siebes, and S. Swift, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 116–126.

[71] Y.-F. Lin, C.-F. Huang, and V. S. Tseng, "A novel methodology for stock investment using high utility episode mining and genetic algorithm," *Applied Soft Computing*, vol. 59, pp. 303–315, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494617302958

[72] L. Wan, L. Chen, and C. Zhang, "Mining dependent frequent serial episodes from uncertain sequence data," in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 1211–1216.

[73] ——, "Mining frequent serial episodes over uncertain sequence data," *ACM International Conference Proceeding Series*, 03 2013.

[74] K. Iwanuma, Y. Takano, and H. Nabeshima, "On anti-monotone frequency measures for

extracting sequential patterns from a single very-long data sequence," in *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, vol. 1, 2004, pp. 213–217 vol.1.

[75] J. Wu, L. Wan, and Z. Xu, "Algorithms to discover complete frequent episodes in sequences," in *New Frontiers in Applied Data Mining*, L. Cao, J. Z. Huang, J. Bailey, Y. S. Koh, and J. Luo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 267–278.

[76] G. Casas-Garriga, "Discovering unbounded episodes in sequential data," in *Knowledge Discovery in Databases: PKDD 2003*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 83–94.

[77] X. Ao, P. Luo, C. Li, F. Zhuang, and Q. He, "Discovering and learning sensational episodes of news events," *Information Systems*, vol. 78, pp. 68–80, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437916303520

[78] T. Katoh, S.-i. Tago, T. Asai, H. Morikawa, J. Shigezumi, and H. Inakoshi, "Mining frequent partite episodes with partwise constraints," in *New Frontiers in Mining Complex Patterns*, A. Appice, M. Ceci, C. Loglisci, G. Manco, E. Masciari, and Z. W. Ras, Eds. Cham: Springer International Publishing, 2014, pp. 117–131.

[79] H. Li, S. Peng, J. Li, J. Li, J. Cui, and J. Ma, "Once and once+: Counting the frequency of time-constrained serial episodes in a streaming sequence," *Information Sciences*, vol. 505, pp. 422–439, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519307169

[80] B. Cule, B. Goethals, and C. Robardet, "A new constraint for mining sets in sequences." in *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*, vol. 1, 04 2009, pp. 317–328.

[81] X. Li, Y. Zhao, D. Li, and W. Guan, "Error serial episodes discovery from mobile payment log in distributed etc," in *Algorithms and Architectures for Parallel Processing*, Y. Lai, T. Wang, M. Jiang, G. Xu, W. Liang, and A. Castiglione, Eds. Cham: Springer International Publishing, 2022, pp. 210–221.

[82] H. Zhu, P. Wang, W. Wang, and B. Shi, "Stream prediction using representative episode

rules," in *2011 IEEE 11th International Conference on Data Mining Workshops*, 2011, pp. 307–314.

[83] S. Laxman, P. Sastry, and K. Unnikrishnan, "Discovering frequent episodes and learning hidden markov models: a formal connection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1505–1517, 2005.

[84] M. Amiri, L. Mohammad-Khanli, and R. Mirandola, "A new efficient approach for extracting the closed episodes for workload prediction in cloud," *Computing*, vol. 102, no. 1, p. 141–200, jan 2020. [Online]. Available: https://doi.org/10.1007/s00607-019-00734-3

[85] G. Liao, X. Yang, S. Xie, P. S. Yu, and C. Wan, "Mining weighted frequent closed episodes over multiple sequences," *Tehnicki Vjesnik-technical Gazette*, vol. 25, pp. 510–518, 2018.

[86] N. Tatti and B. Cule, "Mining closed episodes with simultaneous events," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1172–1180. [Online]. Available: https://doi.org/10.1145/2020408.2020589

[87] B. Tatti, Nikolaj & Cule, "Mining closed strict episodes," *Data Mining and Knowledge Discovery*, vol. 25, pp. 34 – 66, 2012. [Online]. Available: https://doi.org/10.1007/s10618-011-0232-z

[88] P. Fournier-Viger, M. S. Nawaz, Y. He, Y. Wu, F. Nouioua, and U. Yun, "Maxfem: Mining maximal frequent episodes in complex event sequences," in *Multi-Disciplinary Trends in Artificial Intelligence: 15th International Conference, MIWAI 2022, Virtual Event, November 17–19, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 86–98.

[89] N. Tatti, "Significance of episodes based on minimal windows," in *2013 IEEE 13th International Conference on Data Mining*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2009, pp. 513–522.

[90] P. Fournier-Viger, Y. Yang, P. Yang, J. C. W. Lin, and U. Yun, "Tke: Mining top-k

frequent episodes," in *Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices*, H. Fujita, P. Fournier-Viger, M. Ali, and J. Sasaki, Eds. Cham: Springer International Publishing, 2020, pp. 832–845.

[91] S. Rathore, S. Dawar, V. Goyal, and D. Patel, "Top-k high utility episode mining from a complex event sequence," in *COMAD*, 2016.

[92] R. Bathoorn, M. Welten, M. Richardson, A. Siebes, and F. J. Verbeek, "Frequent episode mining to support pattern analysis in developmental biology," in *Pattern Recognition in Bioinformatics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 253–263.

[93] D. Patnaik, S. Laxman, B. Chandramouli, and N. Ramakrishnan, "Efficient episode mining of dynamic event streams," in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 605–614.

[94] J. C. C. Tseng, J.-Y. Gu, P. F. Wang, C.-Y. Chen, C.-F. Li, and V. S. Tseng, "A scalable complex event analytical system with incremental episode mining over data streams," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 648–655.

[95] M. Narmatha and S. H. A. .K, "An online malicious attack detection using honey pot and episode mining," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, 2016.

[96] S. Laxman, V. Tankasali, and R. W. White, "Stream prediction using a generative model based on frequent episodes in event sequences," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 453–461. [Online]. Available: https://doi.org/10.1145/1401890.1401947

[97] X. Ao, P. Luo, C. Li, F. Zhuang, and Q. He, "Online frequent episode mining," in *2015 IEEE 31st International Conference on Data Engineering*, 2015, pp. 891–902.

[98] Z. Hu, W. Liu, and H. Wang, "Mining both frequent and rare episodes in multiple data streams," in *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2013, pp. 753–761.

[99] J. Qin, J. Wang, Q. Li, S. Fang, X. Li, and L. Lei, "Differentially private frequent

episode mining over event streams," *Engineering Applications of Artificial Intelligence*, vol. 110, p. 104681, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622000112

[100] T. You, Y. Li, B. Sun, and C. Du, "Multi-source data stream online frequent episode mining," *IEEE Access*, vol. 8, pp. 107 465–107 478, 2020.

[101] Y.-F. Lin, C.-F. Huang, and V. S. Tseng, "A novel methodology for stock investment using high utility episode mining and genetic algorithm," *Applied Soft Computing*, vol. 59, pp. 303–315, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494617302958

[102] D.-P. Chen and X.-S. Zhang, "Internet anomaly detection with weighted fuzzy matching over frequent episode rules," in *2008 International Conference on Apperceiving Computing and Intelligence Analysis*, 2008, pp. 299–302.

[103] L. Fahed, A. Brun, and A. Boyer, "Deer: Distant and essential episode rules for early prediction," *Expert Systems with Applications*, vol. 93, pp. 283–298, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417307182

[104] ——, "Influencer events in episode rules: A way to impact the occurrence of events," *Procedia Computer Science*, vol. 60, pp. 527–536, 2015, knowledge-Based and Intelligent Information and Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[105] B. Cule, N. Tatti, and B. Goethals, "Marbles: Mining association rules buried in long event sequences," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 7, no. 2, pp. 93–110, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11199

[106] S. Moens, O. Jeunen, and B. Goethals, "Interactive evaluation of recommender systems with sniper: An episode mining approach," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 538–539.

[107] Z. Liu, H. Cai, H. Yu, B. Shen, and L. Jiang, "Constructing the sequential event graph

for event prediction towards cyber-physical systems," in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2021, pp. 1292–1297.

[108] Z. Sun, A. Duncan, Y. Kim, and 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*, vol. 125, p. 103372, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0149197020301244

[109] B. Biswal, A. Duncan, and Z. Sun, "Ada: Advanced data analytics methods for abnormal frequent episodes in the baseline data of isd," *Nuclear Engineering and Technology*, vol. 54, no. 11, pp. 3996–4004, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1738573322003199

[110] M. Zhu, H. Yu, Z. Liu, B. Shen, L. Jiang, and H. Cai, "An intelligent collaboration framework of iot applications based on event logic graph," *Future Generation Computer Systems*, vol. 137, pp. 31–41, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2200228X

[111] "Infrabel open data portal," https://opendata.infrabel.be/pages/home Last Accessed 19 May 2023.

[112] O. Oualid, "Synthetic datasets used by nonepi," https://github.com/Oualidinx/NONEPI-synthetic-data.git, accessed 8 June 2024.

[113] ——, "Datasets used by emdo," https://github.com/Oualidinx/EMDO-synthetic-datasets.git, accessed 8 June 2024.

[114] ——, "Datasets used by uemdo," https://github.com/Oualidinx/UEMDO-datasets.git, accessed 8 June 2024.

[115] P. Fournier-Viger, "Spmf homepage," http://www.philippe-fournier-viger.com/spmf/, accessed 01 Nov 2021.