Applied Computing and Intelligence

http://www.aimsciences.org/journal/aimsaci

Applied Intelligence and Computing
3(1): 44–78
DOI: 10.3934/aci.2023004
Received: 14 September 2022
Revised: 05 January 2023
Accepted: 11 January 2023
Published: 06 February 2023

Survey

From basic approaches to novel challenges and applications in Sequential Pattern Mining

Alessio Bechini[1], **Alessandro Bondielli**[2,3,*], **Pietro Dell'Oglio**[1] **and Francesco Marcelloni**[1]

[1] Dept. of Information Engineering, University of Pisa, Italy; alessio.bechini@unipi.it, pietro.delloglio@unifi.it, francesco.marcelloni@unipi.it

[2] Dept. of Computer Science, University of Pisa, Italy; alessandro.bondielli@unipi.it

[3] Dept. of Philology, Literature and Lingustics, University of Pisa, Italy; alessandro.bondielli@unipi.it

* **Correspondence:** Email: alessandro.bondielli@unipi.it.

Academic Editor: Pasi Fränti

Abstract: Sequential Pattern Mining (SPM) is a branch of data mining that deals with finding statistically relevant regularities of patterns in sequentially ordered data. It has been an active area of research since mid 1990s. Even if many prime algorithms for SPM have a long history, the field is nevertheless very active. The literature is focused on novel challenges and applications, and on the development of more efficient and effective algorithms. In this paper, we present a brief overview on the landscape of algorithms for SPM, including an evaluation on performances for some of them. Further, we explore additional problems that have spanned from SPM. Finally, we evaluate available resources for SPM, and hypothesize on future directions for the field.

Keywords: pattern mining; data mining; process mining; itemset mining; episode mining

1. Introduction

Process mining is a family of techniques that emerged in recent years for the analysis of processes based on event logs [98]. Process mining aims to generate, out of the observation of process logs, factual knowledge possibly useful in several application contexts, supporting performance monitoring tasks, improved exploitation of the available resources and, in particular, the automation of processes themselves.

The input data of a process mining application is called an *event log*, i.e., a collection of chronologically ordered records of events produced by the execution of a process, such as a business

one, where specific nomenclatures and descriptions are typically used [67]. Each event in the log refers to: i) a specific process instance, ii) an activity in such a process, along with related information, and iii) a timestamp. Event logs can usually be obtained from any information system that supports operational processes, from specific software monitoring applications, and from services at the operating system level as well. Event logs can be pre-processed to have at our disposal the essential information as formal sequences of events [67].

From a practical standpoint, and from a more general perspective, the goal of a process mining algorithm is the identification, within a dataset (for example a number of event logs), of particular sequences that show up a statistically significant number of times, in order to verify particular potentially interesting recurrences [45]. The problem of the identification of such sequences has been explored in the literature under different lights and different domains, leading to the classic research problems known as *Frequent Itemset Mining* (FIM) and *Sequential Pattern Mining* (SPM), along with various other derived sub-problems, such as the so-called *High-Utility Sequential Pattern Mining* (HUSPM) and *Frequent Episode Mining* (FEM).

A pattern mining approach can be applied for data analysis in a large number of applications, especially in data explorations that cannot take advantage of prior knowledge on the target problem. FIM was first developed as a tool for market basket analysis to study the behaviour of customers by considering what products are frequently bought together [2]. Any single purchasing event, or *transaction*, can be modeled as a set of items, usually called *itemset*. The availability of databases containing transactions led to the development of *Association Rule Mining*, which consists of finding association rules in a transaction database [2,55]. SPM emerged as a research field addressing problems where itemsets are a suitable means to describe events but, additionally, it is crucial taking into account their sequential ordering. In this setting, databases with sequences of itemsets can be analyzed with techniques in the field of *Sequential Rule Mining*, with the purpose of discovering sequential rules, which are similar to association rules except that they takes into account the sequential ordering [29].

SPM has been exploited in a vast number of applications, including more refined market basket analyses, as well as more complex and demanding tasks like intrusion detection from server logs: intrusion patterns such as brute force attempts at cracking a password can be uncovered by identifying frequent sequences of certain operations.

Extensions on the SPM problem, such as HUSPM and FEM, have proven to be quite useful in different contexts. For example, HUSPM includes utility as an additional dimension of the problem. In the case of market basket analysis, this leads us to consider the profit generated by each specific transaction, and to rank sequential patterns also according to this criterion, considering also the economical perspective for the estimation of their importance [95]. FEM turns to be particularly suited in cases where the system activity is described by a single long sequence rather than a collection of different sequences. A typical example is represented by the activity log of a video-terminal, which tracks the actions taken by users: pattern mining can be leveraged, for example, to mine recurring patterns of usage that can be exploited in the context of robot process automation [18, 21].

Some literature reviews exist around this subject, describing algorithms and approaches for SPM. The two main surveys of the field focused primarily on the classical algorithms and techniques known at the time [32, 70] and, in the second of the two, with a clear didactic objective. Several other works related to the topic have been proposed, but with different perspectives. Significant examples are a complete review of Parallel SPM [37], and an exhaustive 25 years review of FIM [62]. However, the

continuous advancements in the field shed new light both on perspective advancements as well as on the role of more established methods. In this paper we present a consistent account of the approaches proposed in the literature that are related to SPM, along with a brief experimental analysis aimed at characterizing their efficiency and effectiveness. With the aim of providing a more comprehensive overview of the field, some techniques related to SPM have been covered as well, focusing specifically on FEM, a current promising line of research. In addition to this, we explore available resources for SPM.

We must notice that several problems in data mining and machine learning require to operate on sequential data, such as traffic, stock market and weather-related data to name a few. Even if several types of investigations can be carried out on sequential data by means of pattern mining methods like the ones described in this paper, it is important to underline that in general they do not address *predictive* aspects, but deal instead with *descriptive* aspects such as spotting out data regularities (expected or not), thus contributing to a deeper comprehension of the target phenomenon. Other approaches, belonging for example to the fields of Machine Learning and Time Series Forecasting, can be used in prediction problems.

The paper is organised as follows. The basic concepts underlying SPM are introduced in Section 2. In Section 3, an overview of the different types of algorithms for SPM is proposed. Section 4 describes some research fields closely related to SPM. Among them, FEM is particularly important and it is covered in Section 5. Notable resources and open-source software available for SPM are presented in Section 6. Finally, Section 7 outlines possible future perspectives on SPM, and Section 8 draws proper conclusions.

2. Fundamentals of Sequential Pattern Mining

SPM is a branch of data mining that deals with finding statistically relevant regularities among data samples whose values are expressed through ordered sequences. SPM is based on FIM, which was originally introduced in Market Basket Analysis to identify sets of products that are frequently bought together [2]. The problem studied by FIM does not explicitly consider the temporal ordering of records to be analyzed, and it is defined as follows.

Let us consider a *transactional* database D, consisting in a set of transactions $\{t_i\}$. Any single transaction, i.e. the basic record in the database, is typically identified by a Transaction ID (TID) and corresponds to a set of objects (items or symbols) out of a set of possible items $I = \{i_1, i_2, ..., i_n\}$. For example, regarding the sales in a shop, each transaction represents a receipt, and thus the items purchased by a particular customer. In this case, I is the set of all the items sold by the shop. A subset $X \subseteq I$, containing k items, is called an *itemset*.

It should be noted that no constraint applies to the data types of the elements in the set I, which can be also heterogeneous. In practice, several algorithms (for both FIM and SPM) ask to operate over I and to scan the relative itemsets according to a total order over their members (for example, the lexicographic order); thus, often such items are kept in the supporting data structures in the order required. The proportion of transactions in D that contain the itemset X is called the *support* of X, typically denoted as $supp(X, D)$ (or just $supp(X)$, whenever D is implicitly given). To consider an itemset as frequent within D, and thus of interest, it must exceed a minimum support threshold *minsupp*. An itemset X is therefore frequent if $supp(X) \geq minsupp$ in D. The support, and consequently the

minimum support threshold, can take values in the range $[0, 1]$. Typically, in FIM algorithms it is possible to specify the actual value for the *minsupp* parameter, so that more (lower *minsupp*) or fewer (higher *minsupp*) elements would be considered as frequent. The objective of FIM is thus defined as the identification of all the itemsets within a specific transaction database whose support is at least *minsupp*.

Whenever the temporal ordering of transactions is relevant for the analysis goals, different approaches must be devised. SPM has been proposed as an extension of FIM to handle also temporally ordered sequences of itemsets. The problem studied by SPM is defined as follows. A *sequence* database $SeqD = \{S_1, S_2, ..., S_j\}$ is made of j *sequences*, and each of them is supposed to be generated by a distinct "source." A sequence is identified by a Sequence ID (SID) and it consists of an *ordered list* of elements: in the most general case, an element is an itemset (usually, a non-empty one). In the shop example, a sequence could represent the successive different visits to the shop of a given customer. Each visit corresponds to a transaction, and therefore has a corresponding itemset, which, as in the previous case of FIM, may contain one or more objects (items or symbols).

Usually, the ordered elements of the sequence (i.e., the itemsets) are referred to as *events*. Given a sequence $S = \langle E_1, E_2, ..., E_n \rangle$, the corresponding length $|S|$ can be defined as the number of events in S. On the basis of the total number of objects it contains, a sequence is referred to as a k-sequence, with $k = \sum_{i=1}^{n} |E_i|$. Consider the example sequence $s = \langle \{milk, cereals\}, \{beer, bread, meat\}, \{detergent\} \rangle$. The sequence contains three events, i.e. three distinct visits of a customer to the shop. The length of the sequence is therefore $|S| = 3$, and it is a 6-sequence containing $k = 2 + 3 + 1 = 6$ purchased items. Practical examples of sequence databases are available in [32].

The extension of the concept of *support* to sequences requires the introduction of some additional definitions. A sequence s_2 is said to be a *sub-sequence* of the sequence s_1 if it can be derived from s_1 by deleting some objects without changing the order of the remaining objects [32]. Formally, $s_2 = \langle B_1, B_2, ..., B_n \rangle$ is a sub-sequence of $s_1 = \langle A_1, A_2, ..., A_m \rangle$ if and only if there exist n integers $\{h_i\}_{i=1...n}$ with $1 \leq h_1 < h_2 < \cdots < h_n \leq m$ such that $B_1 \subseteq A_{h_1}, B_2 \subseteq A_{h_2}, ...B_n \subseteq A_{h_n}$. This is indicated by the notation $s_2 \sqsubseteq s_1$, and we can say that s_1 contains s_2 or, likewise, s_1 is a *super-sequence* of s_2. The definition of sub-sequences let us generalize the notion of "support": the support of a sequence s within the database $SeqD$ is the proportion, over the size of $SeqD$, of those sequences that contain s, i.e. $supp(s, SeqD) = |\{s' \mid s' \in SeqD \wedge s \sqsubseteq s'\}| \; / \; |SeqD|$. It must be noted that sometimes in the literature the support is considered the plain count of super-sequences of S in $SeqD$.

Given a minimum support threshold *minsupp*, a sequence s in a sequential database $SeqD$ is called a *frequent sequential pattern* (or "sequential pattern" for short) if $supp(s) \geq minsupp$. The objective of SPM is therefore the identification of the sequential patterns within a given database of sequences [1], i.e. its *SPM set*.

Tables 1 and 2 show, respectively, an example sequence database and the sequential patterns mined from it for a specific value of threshold *minsupp*. For the sake of generality, in the example we adopted "placeholders" for the item names (a, b, c, and so on). Thus, such a general sequence database can represent a large number of cases, depending on what the items refer to: transactions made by customers in a store, points of interest visited by groups of tourists in a city, interactions of a user with a computer, and so on. We used 7 items, that are included in 11 distinct itemsets (denoted by curly brackets), with events distributed across 4 different sequences of assorted lengths.

The goal of SPM is to find the *patterns of events* that occur in the dataset with a support greater

Table 1. An example of a sequence database (horizontal format).

Seq. ID	Sequence of events (itemsets)
1	{a,c}, {b,c}, {c}, {d,e,f},{g}
2	{a,b},{c}, {f}, {d,e,g}
3	{a},{b}, {f,g},{c}, {b,c}, {d,e,f}
4	{b},{f,g}

than the threshold *minsupp*. In our case, we set *minsupp* to 2. Thus, all the patterns of events with a support equal or greater than 2 are taken as *sequential patterns*. In order to determine them, it is sufficient to compute the support (i.e., how many times the actual pattern appears in the sequences of the database) for each sequence and sub-sequence of the database. Table 2 shows the support values for several sequences of events: in particular, sequential patterns are indicated in bold.

Table 2. A subset of sequences from the database in Table 1 and their support. Sequential patterns with *minsupp* ≥ 2 are indicated in bold.

Pattern	Support
{c}	**3**
{b}	**2**
{b,c}	**2**
{d,e,f}	**2**
{b},{f,g}	**2**
{a},{b},{f,g}	1
{a,c},{b,c}	1
{c},{f},{d,e,g}	1
...	...

SPM is a complex and computationally demanding problem, and this is not surprising, just considering the simple fact that a k-sequence has 2^k sub-sequences, which is a huge search space for SPM algorithms. Typically, SPM algorithms explore the search space by generating new candidates (i.e., new sequences), and identifying their support within the database. Candidate generation is typically accomplished in two ways: sequence extension (*s-extension*), and itemset extension (*i-extension*). In an s-extension, an S_p sequence is extended by adding a new event containing a single item, while in an i-extension a new item is added to the last S_p event.

As far as specific search approaches are concerned, there are typically two categories of algorithms for SPM, namely *depth-first* and *breadth-first*. With breadth first algorithms, the database is typically scanned for sequential patterns of increasing length. First sequence patterns of length 1 are identified, then length 2, up to a length k which determines the maximum sequence length in the database. As for depth-first algorithms, they typically organize the search space in a prefix tree structure. The first level of the tree includes sequences with single events. Then, their children are generated via s-extensions and i-extensions, until no more candidates can be generated. Finally, the support of all the generated sequences is computed with respect to the actual data in order to identify the frequent

sequential patterns [32].

3. Algorithms for Sequential Pattern Mining

A large number of algorithms and implementations have been proposed in the literature for SPM, and it has been proposed to group up them in few broad classes on the basis of the way the task is performed [70]. For the sake of description clearness, in this paper we roughly follow the categorisation recently adopted by Gan et al. [37], which comprises four different groups of SPM algorithms: *apriori-based*, *pattern-growth*, *hybrid*, and *constraint-based*. In brief, apriori-based algorithms exploit the antimonotonicity of the apriori property to prune infrequent sequences. Pattern growth algorithms generally work by exploiting a projected database based on prefixes to count only occurrences of actual patterns in the database. Hybrid algorithms incorporate notions from both apriori-based and pattern growth ones to leverage their strengths and reduce their drawbacks. Finally, constraint-based algorithms focus on a more specific problem, i.e. finding frequent patterns that satisfy certain constraints, regardless of the used method. A graphical representation of the types of algorithms and their relations is depicted in Figure 1. In this section we will discuss all of these categories, and highlight the best or more popular implementations for each one.

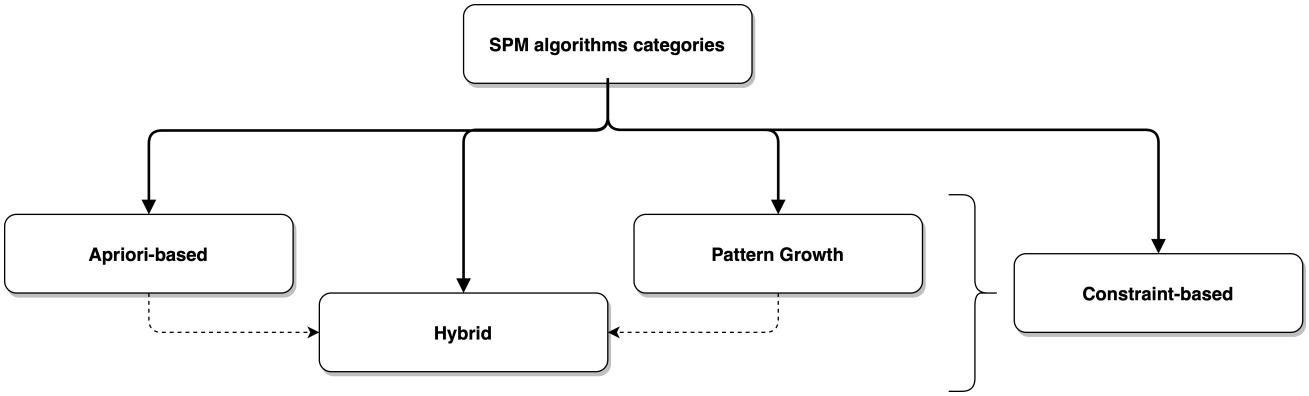

Figure 1. Types of algorithms for SPM and their relationships.

It is worth underlining that we are currently focusing our attention on *exact* discovery of frequent sequential patterns, i.e., all the ones whose support value is greater than or equal to the provided *minsupp* (i.e., the minimum support threshold), which is a hyper-parameter for the algorithm. Thus, given a sequence database and a value for *minsupp* (and possibly some constraints in specific algorithms), any distinct algorithm must return exactly the same set of frequent sequential patterns. All the proposed techniques may differ in the adopted computational approach, leading to different execution times and memory footprints, but the identified sequential patterns will always be the same [32]. Generally, the application of specific ideas and mechanisms in each algorithm requires the adoption of proper data structures, designed on-purpose [70]. A summary of the main algorithms described hereafter is reported in Table 3, which does not include those relative to specific re-formulations of the SPM problem (mainly, constraint-based algorithms, which are summarized in Table 5).

Table 3. Algorithms for SPM.

Name	Type	Description	Year
Apriori [1]	Apriori-based	The first approach SPM	1995
GSP [93]	Apriori-based	Generalized SPM algorithm	1996
SPADE [120]	Apriori-based	Using equivalence classes to discovery sequential patterns	2001
LAPIN-SPAM [112] and LAPIN [113]	Apriori-based	SPM with last position induction	2004-2005
FreeSpan [43]	Pattern Growth	Using frequent pattern-projected to discover sequential patterns	2000
WAP-Tree [74]	Pattern Growth	SPM for web logs with the WAP-tree data structure	2000
PrefixSPAN [44]	Pattern Growth	Prefix-projected SPM	2001
FS-Miner [23]	Pattern Growth	SPM with suffix growth	2004
DFSP [57]	Pattern Growth	Mining frequent biological sequences	2014
HVSM [89]	Hybrid	First-Horizontal-last-Vertical scanning database SPM algorithm	2005
DISC-ALL [17]	Hybrid	Position induction and prefix growth SPM	2007
UDDAG [16]	Hybrid	Up-down directed acyclic graph for SPM	2010

3.1. Apriori-based algorithms

Apriori-based algorithms exploit a general property of sequences, known as the *apriori property*, which states that "all non-empty subsets of a frequent itemset must also be frequent" (see for example [64]). This means that, in turn, given an infrequent itemset, all of its supersets (i.e., the itemsets that contain such itemset) will also be infrequent: this gives us the practical opportunity to properly bound the exploration of the search space. Apriori-based algorithms typically use a breadth-first search for sequences within the database.

Apriori-based algorithms can exploit *horizontal* and *vertical* formats for the database, which are alternative ways to represent the same content. The former is the standard database representation, as shown in the example in Table 1. The vertical representation of a sequence database is made of the *IDList*s of all the single items: each IDList is relative to a specific item, and reports the sequences where such an item appears and, for such sequences, in what events (itemsets) it is present. The vertical representation of the dataset in Table 1 is presented in Table 4, where IDLists are reported in columns; e.g., considering the IDList for item *b*, we can see that it appears in sequence 1 (event 2), in sequence 2 (event 1), in sequence 3 (events 2 and 5), and in sequence 4 (event 1). The support of a pattern can be easily and directly calculated by checking the content of the IDLists of the pattern items.

An overview of apriori algorithms for SPM is presented below.

Apriori The first approach for SPM based on the apriori property was proposed by Agrawal and Srikant [1]. It requires the generation of *candidate sequences*, and to check them against the database: It makes use of the antimonotonicity of the apriori property to avoid unnecessary checks. Following the definition, if a sequence a of length n is infrequent (i.e., its support is below the minimum support threshold), then all sequences that contain a will also be infrequent, and can therefore be discarded a priori in the search. Thus, by exploiting this property, it is possible to reduce the search

Table 4. Vertical representation of the sequence database in Table 1. IDLists for the items are represented by the table columns.

	Item IDLists						
Seq. ID	**a**	**b**	**c**	**d**	**e**	**f**	**g**
1	1	2	1, 2, 3	4	4	4	5
2	1	1	2	4	4	3	4
3	1	2, 5	4, 5	6	6	3, 6	3
4	-	1	-	-	-	2	2

space disregarding all the super-sequences of infrequent sequences. From an implementation point of view, the authors propose three different variants of the algorithm: *AprioriAll*, *AprioriSome*, and *DynamicSome*. They all exploit a breadth-first approach. Thus, they have to perform several passes over the entire database of sequences to extract the frequent ones. In the first pass, candidate sequences of unit length are identified, which occur at least *minsupp* times. Next, sequences of length 2 that also contain the candidate sequences of length one, identified in the first pass, are analysed, and so on. In AprioriAll, all frequent sequences are calculated at each step and the maximum length sequences are retained for use in the next step. In AprioriSome, on the other hand, the longest frequent sequences are calculated at each step, so that all the shorter sequences that are their sub-sequences can be discarded. Finally, DynamicSome is similar to AprioriSome, but at each step the frequent sequences are generated "on the fly," i.e., considering the frequent sequences in the previous steps and those obtained by scanning the dataset [1].

GSP The *Generalized Sequential Pattern* algorithm (or GSP [93]) exploits the apriori property as well, addressing performance improvements over AprioriAll and introducing also three novelties for better identifying interesting frequent sequences: time constraints (minimum and maximum gap between transactions), sliding windows, and taxonomies. The time constraints between adjacent elements aim to avoid considering adjacent elements that are too close to each other or too far apart in time. Further, the sliding window mechanism allows for items in an element of a sequential pattern to come from different transactions. Items that are present in a set of transactions within a user-specified window can belong to the same pattern. Finally, user-defined taxonomies have the purpose to better describe the items in the database. Aside from these key aspects, GSP is very similar to AprioriAll, and proceeds with successive steps. First, k-sequences are identified for $k = 1$. Then, at each step, candidate $k + 1$-sequences are generated and counted in the database, in order to prune infrequent ones for subsequent steps. The empirical evaluations presented by the authors of GSP indicate that it outperforms AprioriAll. Further improvements in efficiency have been obtained in PSP, a version of GSP that makes use of a different supporting data structure [68]. GSP uses a horizontal representation of the dataset.

SPADE All the algorithms presented so far proceed through successive multiple scans of the input dataset: A possible reduction of the number of the required scans is expected to provide significant

computational advantages. The *SPADE* algorithm [120] is certainly one of the most widely used SPM algorithms that exploit the apriori property, and it has been designed to overcome the need to scan the database several times. SPADE decomposes the problem into sub-problems using a set of combinatorial properties. The final result can thus be achieved with no more than three scans of the database. SPADE uses a vertical representation of the dataset.

The SPAM algorithm [6] is a depth-first algorithm very similar to SPADE: its main difference is that it makes use of a vertical representation of the database rather than a horizontal one.

LAPIN-SPAM and LAPIN The LAPIN-SPAM algorithm [112] provides a number of improvements over the approach proposed for SPAM [6]. The LAPIN algorithm [113], proposed a couple of years later, uses a technique called *last position induction*. The idea is to exploit the last position of an object i within the database to decide whether or not a k-sequence can be extended using i.

3.2. Pattern growth algorithms

Most apriori-based techniques typically rely on identifying patterns by means of insertion and deletion of new objects or events from the shortest patterns, and identifying their frequency in the database. This is a rather costly approach, as at each iteration candidates are first generated and then counted throughout the dataset. Pattern-growth algorithms attempt to address this limitation by incrementally building up the patterns actually present in the database. A recursive approach is yet quite costly. Thus, the use of a projected database was proposed, i.e., a compressed representation of the original database composed of projected itemsets [44]. The idea behind this approach is that, to test whether an itemset X is frequent, we can consider the X-projected database, i.e. a sub-database that includes only transactions in which X appears. Following from this, and based on the assumption that if an itemset X is infrequent, then any sequence whose projected itemset is a superset of X cannot be a sequential pattern, we can prune large chunks of the database and exploit only the projected sub-databases to perform further searches.

Pattern growth algorithms typically use a depth-first search of sequences within the database.

The following is an overview of the most commonly used SPM algorithms based on pattern growth.

FreeSpan One of the first proposed algorithms to exploit the idea of a projected sequence database has been *FreeSpan* [43]. The idea behind FreeSpan is to use the frequent items to recursively project the sequence databases into smaller ones, and use such databases to grow the subsequence fragments. Thus, each frequent pattern to be tested is confined to its corresponding, smaller, projected database [43].

PrefixSPAN Although its proposal dates back to the early 2000s, *PrefixSPAN* [44] is still one of the pattern growth algorithms most widely used today. It initially identifies all frequent sequences consisting of a single event with a single item. Given for example frequent items a, b, c, d, the algorithm analyses all and only those sequences whose prefix is one of the frequent items, through a depth-first search. Thus, the algorithm analyses all the sequences of the type $\langle\{a\}, ..\rangle$, then those of the type $\langle\{b\}, ..\rangle$ and so on. In this way, the need to iterate over the entire dataset several times is avoided. The authors

have shown that the algorithm performs better than both GSP and FreeSpan. PrefixSPAN has also been shown to perform better on larger databases [37]. PrefixSPAN-x is a variant of PrefixSPAN that keeps its overall structure, introducing further artifices [26]. The idea behind the algorithm is to make it more efficient, both in terms of time and memory, by removing all infrequent objects from the database.

DFSP The *Depth-First SPelling* (DFSP) algorithm [57] has been specifically designed for mining frequent biological sequences such as DNA and, on this particular task, it performs better than PrefixSPAN.

FS-Miner Web logs represent an interesting target for frequent sequence mining, and *FS-Miner* is an algorithm developed to operate in this specific setting [23]. The algorithm scans the database twice: In the first place, it identifies the support of sub-sequences of length two; Subsequently, other potentially frequent super-sequences are identified starting from these. These are then represented in a tree structure (FS-Tree) that simplifies the mining operations for the identification of frequent patterns.

WAP-Tree Similarly to FS-Miner, *WAP-Tree* [74] uses a tree structure and requires only two scans of the database. However, in this case only the support of sequences of length one is calculated during the first scan. Once the tree is generated, it is recursively scanned to identify frequent super-sequences. Several improvements over WAP-tree have been proposed in the literature, including PLWAP [24], a version that does not require to recursively scan the tree.

3.3. Hybrid algorithms

In addition to apriori-based and pattern growth algorithms, other hybrid approaches aim to leverage the strengths of both Apriori-based techniques and pattern growth ones, whereas reducing their downsides. It is in fact true that pattern-growth based algorithms have improved on some critical aspects that hindered the performances of a simpler apriori-like approach. Nevertheless, several issues emerged for pattern growth ones. For example, some algorithms such as PrefixSpan have a high computational cost because of the method used to construct the projected database of sequences. The research on "hybrid algorithms" is therefore aimed at identifying and exploiting the strengths of both apriori and pattern-growth algorithms to optimise the pattern search procedure. Hereafter we overview the most relevant hybrid algorithms for SPM.

HVSM The structure of *HVSM* (first-Horizontal-last-Vertical scanning database Sequential pattern Mining algorithm) [89] follows the typical approach of the SPAM algorithm. In fact, HVSM considers the database as a vertical bitmap. It first extends the itemsets horizontally, extracting all the itemset sequences; subsequently, it extends them vertically, and generates the candidate sequences. Experiments have shown that the HVSM algorithm can search for frequent sequences faster than the SPAM algorithm, especially when using very large sequence and/or transaction databases [89].

DISC-All Disc-All (DIrect Sequence Comparison) [17] allows frequent sequences to be mined without necessarily having to calculate the support of less frequent sequences. This algorithm does not

Table 5. Algorithms for Constraint-based SPM.

Name	Type	Description	Year
SPIRIT [38]	Constraint-based	SPM with regular expression constraints.	2002
CLaSP [41]	Closed	An efficient algorithm for mining frequent closed sequences.	2013
CloFAST [36]	Closed	closed SPM using sparse and vertical IDLists.	2016
BIDE [100]	Closed	efficient mining of frequent closed sequences.	2004
CCSpan [121]	Closed	Mining closed contiguous sequential patterns.	2015
NetNCSP [106]	Closed	Nonoverlapping closed sequential pattern mining.	2020
MSPX [63]	Maximal	Efficient mining of maximal sequential patterns using multiple samples.	2005
MaxSP [28]	Maximal	Mining maximal sequential patterns without candidate maintenance.	2013
VMSP [30]	Maximal	Efficient vertical mining of maximal sequential patterns.	2014
QCSP [27]	Top-K	Mining top-k quantile-based cohesive sequential patterns.	2018
AprioriQSP [52]	Quantitative Sequences	SPM with quantities.	2007
PrefixSpanQSP [44]	Quantitative Sequences	Mining sequential patterns efficiently by prefix-projected pattern growth.	2001
Q-VIPER [20]	Quantitative Sequences	Quantitative vertical bitwise algorithm to mine frequent pattern.	2022

rely on the antimonotonicity property [17]. In fact, it applies an early pruning approach for infrequent sequences, while taking into account other sequences of the same length.

UDDAG UDDAG (UpDown Directed Acyclic Graph) [16] exploits bidirectional pattern growth, using suffixes and prefixes to extract frequent sequences to improve on the performances of other pattern growth-based algorithms, especially in terms of scalability (i.e., at increasing sizes of *minsupp* and for length-k patterns with larger k).

3.4. Constraint-based algorithms

Beside investigations on specific data structures and techniques for more and more efficient identification of frequent sequential patterns, the literature has seen a rather large interest for particular problems that require the introduction of specific constraints for the frequent patterns' retrieval. *Constraint-based SPM algorithms* (CSPM) focus on finding sets of sequence patterns that satisfy a given constraint C. Formally, a constraint C for the SPM task is a predicate $C(\cdot)$ that determines whether a sequence in the SPM set can be considered as acceptable or not [76]. The retrieval procedure must select only the sequences that satisfy the constraint, thus yielding a more compact result, hopefully containing the more interesting sequences.

A general way to express a constraint may be the adoption of regular expressions, as it has been proposed for the family of algorithms known as SPIRIT (Sequential Pattern mIning with Regular expressIon consTraints) [38]. Other types of constraints have been identified [70, 75]: on the item type, on aggregate functions of items, on the pattern length, and model-based constraints (i.e., sub-pattern or super-pattern of a given pattern); whenever events are associated with timestamps, it could be possible to specify time span, time difference between adjacent events, and so on. Moreover, *gap constraints*

have been studied in particular, and they correspond, in the specification of the target pattern, to the possible presence of a number of events (in a given range) between two adjacent, given events [105].

It is often required to meet specific criteria on the number of occurrences of a pattern in a sequence, with respect to any timing constraint that may have been imposed. There exist five different counting techniques that have been classified into three groups by the authors of [70]. The first group consists of the CEVT technique (count event), which searches for the given sequence in the entire sequence timeline. The second group consists of the CWIN (count windows) and CMINWIN (count minimum windows) techniques. They deal with counting the number of windows in which a given sequence occurs. The last group consists of the CDIST (count distinct) and CDIST_O (count distinct with the possibility of overlap) [70].

Often CSPM algorithms leverage specific properties of the imposed constraint, like *monotonicity* (if $C(S)$ is false, then the same result applies to all sub-sequences of S) or *anti-monotonicity* (if $C(S)$ is false, the same holds for all super-sequences of S); in case none of such properties is satisfied, the development of the CSPM algorithms becomes more challenging [47]. CSPM algorithms can be organised into several widely studied subgroups, discussed below. In the following, we focus among others on *closed and maximal sequential pattern mining* (and *sequential generators as well*), *top-k sequential patterns*, and *quantitative sequences* [37]. We summarize constraint-based algorithms in Table 5.

3.4.1. Closed and maximal SPM

Often not all the sequences in an SPM set are relevant to the user, and it would be reasonable keeping only a significant subset of them, considering that if a sequence is frequent, then its sub-sequences are frequent as well. One of the critical issues with the algorithms presented so far is their tendency to extract redundant patterns, especially for very low *minsupp* values, or in presence of pattern-enriched databases [121]. To alleviate this problem, it has been proposed to define sets of Closed Sequential Patterns (CSPs) and Maximal Sequential Patterns (MSPs). A CPS set S_{closed} contains the sequences in the SPM set S_{freq} that have no super-sequences with the same support in it; formally [111], $S_{closed} = \{s \mid s \in S_{freq} \wedge \nexists\ s' \in S_{freq} \text{ s.t. } s \sqsubseteq s' \wedge supp(s) = supp(s')\}$. Obviously, the CPS set is contained in the SPM set. Similarly, the MSP set S_{maxf} contains all the *maximally frequent sequences*, i.e. those in the SPM set with no super-sequences in it [63]; formally, $S_{maxf} = \{s \mid s \in S_{freq} \wedge \nexists\ s' \in S_{freq} \text{ s.t. } s \sqsubseteq s'\}$. Clearly, the MPS set is contained in the CPS set.

Yan et al. [111], along with introducing the definition of CSP set, proposed also the *CloSPAN* (Closed Sequential PAtterN mining) algorithm for determining the closed sequential patterns. By leveraging the founding ideas of both SPADE [6] and CloSPAN [111], the *CLaSP* algorithm [41] exploits a vertical data representation and pruning strategies to improve its efficiency over CloSPAN. Similarly, *CloFAST* [36] uses a vertical database layout and sparse id-lists to identify closed sequential patterns. The *BIDE* algorithm (the acronym stands for "BI-Directional Extension based frequent closed sequence mining") [100] adopts techniques for an effecting pruning of the search space, yet keeping low the memory requirements. Yet another proposal to cope with redundancy in results is represented by *CCSpan* (Closed Contiguous Sequential pattern mining) [121], which is intended to obtain an even more compact set of patterns as a result, yet avoiding any information loss. More recently, the exploitation of a NetTree data structure led to the implementation of the *NetNCSP* (Nettree for Nonoverlapping Closed Sequential Pattern) algorithm [106].

The rationale for pursuing the identification of an MSP set lies in the need to properly prune the result set, avoiding confusion to end users and facilitating the interpretation of the algorithm outcome. Moreover, it is worth noticing that a very large result set could also affect the task performance in terms of both execution time and memory. Among the various proposals specialised in retrieving maximal sequential patterns we can mention *MSPX* [63], which uses multiple samples to exclude all those sequences that are less frequent. *MaxSP* (Maximal Sequential Pattern mining) [28] computes all maximal sequential patterns with no need to store intermediate candidate sequences in main memory. *VMSP* (Vertical mining of Maximal Sequential Patterns) [30] has been the first vertical algorithm for mining maximal sequential patterns, obtaining state-of-the-art performances in terms of execution time.

Beyond closed and maximal sequence patterns, other sequences in the SPM set known as *generator sequential patterns*, have been investigated [60]. We can observe that the elements of the SPM set supported exactly by the same sequences in the database can be considered an equivalence class. Clearly, all the sequences of such an equivalence class have the same support, and the "$\sqsubseteq$" relationship is a partial order over them. According to the "$\sqsubseteq$" ordering, the set of maximal and minimal patterns within an equivalence class are called *closed patterns* and *generator patterns*, respectively. Thus, the set of generator sequential patterns, or GSP set, is the set of sequences in the SPM set that have no subsequence with the same support in it; formally, $S_{gen} = \{s \mid s \in S_{freq} \wedge \nexists\ s' \in S_{freq} \text{ s.t. } s' \sqsubseteq s \wedge supp(s) = supp(s')\}$. It has been argued that the properties of the GSP set can be particularly useful, and algorithms have been proposed to find them [54, 60, 77].

3.4.2. Top-k SPM

For particular cases, whenever a correct minimum support threshold is difficult to be ascertained in advance, or when such a choice can heavily influence the obtainable results (for example, too many patterns or uninformative ones), top-k sequential pattern mining has been proposed. Rather than presenting a final result with all the sequential patterns that satisfy specific conditions of length and minimum support, in this case the algorithms are aimed at returning only the k most frequent sequential patterns within the dataset, ordered according to the relative support [32]. Several variations of this approach have been proposed over the years; among the others, it is worth recalling an algorithm based on constrained prefix-projected pattern growth that exploits a novel interestingness measure, which computes the proportion of the occurrences of a pattern that are cohesive [27]. The Top-k approach is often used in combination with other techniques and in SPM-related problems, such as frequent episode mining [34] (see Section 5), or High-Utility mining [82] (see Section 4.1).

3.4.3. Quantitative sequences

For various applications, each item belonging to the events of sequences can be enriched with a *quantitative attribute*, i.e., the actual quantity appearing in the transaction, with the purpose of supporting the comparison of values [52]: such additional information can be extremely useful to end users. For example, knowing quantities of items in a sales transaction and identifying patterns that include quantitative information may be helpful in designing market campaigns. The extension of SPM algorithms to deal with sequences with quantitative attributes is not straightforward, and proper solutions must be designed to get to more complete, and potentially more informative

results. Two different variants of traditional algorithms for quantitative SPM have been proposed [52], namely *Apriori-QSP*, based on the Apriori-like algorithms, and *PrefixSpanQSP*, based instead on PrefixSpan [44]. *Q-VIPER* addresses the problem in the context of Big Data [20].

3.4.4. Other approaches

Another interesting problem with solutions belonging to the typology of constraint-based algorithms is known as *Location-Item-Time Sequential (LIT) Pattern Mining*, which has been introduced by Tsai et al. [97]. The paradigmatic use case is the description of temporal and spatial behaviour of visitors to some theme parks, which can be obtained by exploiting SPM methodologies. An approach of a completely different type is based on a probabilistic model of the sequence database [35]. The idea is to extract the most relevant patterns and to sort them by exploiting an associative measure of significance (i.e., how much the end user may find them interesting) [35]. Yet another type of constraint-based problem that emerged in recent years is the extraction/search of malicious sequential patterns. In particular, a recently proposed algorithm [25] exploits an All-Nearest-Neighbor classifier built specifically for malware detection tasks based on pattern search.

It is worth recalling a recent approach to further limit the results to the most (likely) important sequences: NTP-Miner [107] uses a strategy where attributes are categorized in three different levels, and gap constraints are considered as well in assessing the relevance of a result sequence. Notably, attempts to efficiently obtain meaningful results have been done also for the problem of Negative SPM, which aims to discover events that should have occurred but have not occurred [108].

Constraint-based strategies are widely used in several application fields. An approach based on mining sequential patterns with flexible constraints of Massive Open Online Course (MOOC) platforms enrollment has been proposed in [90]. In particular, the authors identified three constraints: the *length constraint* to describe the effect of the length of enrollment sequences, the *discreteness constraint* to describe the variance of enrollment dates, and the *validity constraint* for the enrollment times.

3.5. Parallelism in Sequential Pattern Mining

In recent years, the ever-increasing availability of large amounts of complex data has gone hand in hand with the possibility to exploit computational resources across different devices. Hence, research on Parallel SPM (PSPM) has attracted more and more attention [37]. The need to apply SPM techniques to big data pushed towards the adoption of multiple computational units. New solutions would thus resort to algorithms able to analyse and identify sequential patterns in a parallel fashion. In many different fields of data mining, the MapReduce programming model proposed in Hadoop has been a key paradigm for enabling the analysis of big data over distributed platforms [88, 96], and has played a central role in parallel SPM as well. Conversely, the Apache Spark framework, which provides implicit parallelism for big data analysis, has not been widely used for SPM as it has been in other fields of data mining [7].

In the first proposed approaches, a number of algorithms have been designed according to data partitioning schemes [37], and parallelised versions of classical SPM algorithms have been developed. In the field of Apriori-based algorithms, MapReduce implementations have been studied in depth [61]. It is worth recalling a couple of the first proposals: pSPADE [119] and webSPADE [22] have been designed as parallelised versions of SPADE [120]. In pSPADE the search space is subdivided into

small categories based on the suffixes of the sequences, which can then be solved easily by means of search techniques and join operations. webSPADE instead is a parallel approach to the analysis of click streams found within website logs. As web log analysis is becoming more and more important, dedicated solutions to analyze massive quantities of this kind of data via a MapReduce approach have been recently developed [92].

Some parallel/distributed versions have also been proposed for the GSP algorithm (and its variant PSP). DGSP [116] and DPSP [49] are solutions designed according to the MapReduce programming model. PartSpan [80], on the other hand, is a distributed and parallel algorithm and was developed to identify patterns of trajectories. Moreover, GridGSP [103] relies on a grid computing environment to parallelise the GSP algorithm. Variants of the SPAM algorithm are SPAMC [14], which uses the MapReduce framework for distributing the computation, and SPAMC-UDLT [15], which further improves the performance of SPAMC by introducing solutions to address scalability issues, thus making it able to handle extremely large databases.

Several concurrent versions have been proposed for pattern growth-based algorithms as well. In particular, Par-ASP [19] and Sequence-Growth [56] are worth mentioning as parallelised versions of PrefixSPAN.

Finally, the literature has also focused on parallelised hybrid algorithms, such as MG-FSM [69] and MG-FSM+ [8]. LASH [8] is another hybrid parallel algorithm that exploits hierarchies. Other parallelised hybrid algorithms include Distributed SPM [39], which exploits dynamic programming and an extended prefix-tree to store intermediate results; ISM (Interesting Sequence Miner) [35], a novel algorithm based on a probabilistic model and exploiting machine learning methodologies, and ACME, [86] which, instead, is a combinatorial method for extracting patterns from a long single sequence generally used in bioinformatics [37].

3.6. Experimental analysis

Generally speaking, a proper choice of an algorithm for one specific application context can be taken only upon the knowledge of the algorithm's behavior with respect to the basic characteristics of the target dataset. To this aim, we report in this section the results of some experimental analyses for the assessment of effectiveness and efficiency of four representative algorithms out of those previously described. In particular, in the first place we want to evaluate the average execution time and the used memory in a typical usage setting. Subsequently, we want to characterize the scalability of the four algorithms in terms of their behavior with respect to the size of the target dataset. All the experiments have been carried out on commodity hardware and software, i.e. an ordinary desktop with an Intel(R) Core(TM) i7-8565U CPU 1.80 GHz, 8.00 Gb of RAM, with MS Windows 11 as operating system, and using Java 8.

Concerning the algorithms we selected GSP, SPADE, Prefix-Span, and Clo-Span, as they can be considered representative of the various groups described in Section 3, and because it is possible for them finding implementations, all in the Java language, within one single library available on the web (SPMF, see Section 6), thus avoiding problems deriving from comparing implementations over extremely diverse frameworks. We tested these algorithms on three datasets, named Bible, MSNBC and Bike, whose characteristics are briefly summarized in Table 1. Such specific datasets have been selected because they present an adequate variety in the total number of sequences, the number of items, and average sequence lengths. Moreover, they refer to extremely different contexts. Bible is

a conversion of the Bible into a sequence database (each word is an item). The dataset is available on the mentioned SPMF library. MSNBC is a dataset of click-stream data from the MSNBC website, converted from original data from the UCI repository. It is available in the SPMF library as well. Bike contains sequences of locations where shared bikes have been parked in the city of Los Angeles. Each item represents a bike sharing station and each sequence indicates the different locations of a bike over time. It has been obtained from the Github of Andrea Tonon, and it is a transformation of a dataset on the popular data science website Kaggle*.

Table 6. Datasets used in the experiments.

	Bible	**MSNBC**	**Bike**
Sequence count	36,369	989,818	338
Item count	13,905	17	17,203
Average sequence length	21.6	13.23	216.44

In the first experimental analysis, for each algorithm and dataset we recorded the number of obtained frequent sequences and basic performance figures, i.e. the execution runtime and the required memory.

In every run for this experiment, the same typical value for *minsupp* has always been used (namely, 0.2), yielding 174 frequent sequence patterns for Bible, 338 for MSNBC, and 23 for Bike. The results about runtimes and memory usage are reported in the two sections of Table 7. We can observe that, regardless of the specific target dataset, SPADE obtained the best results in terms of running time, whereas Prefix-Span was generally the least computationally expensive in terms of memory footprint.

Table 7. Performance of algorithms on a typical usage setting (*minsupp* = 0.2).

	Runtime (ms)			Memory footprint (Mb)		
	Bible	**MSNBC**	**Bike**	**Bible**	**MSNBC**	**Bike**
GSP	5901	17203	1668	298.72	216.44	142.35
SPADE	**603**	**556**	**30**	134.27	375.28	58.48
Prefix-Span	641	602	173	**125.51**	**81.41**	**35.82**
Clo-Span	3135	5896	379	504.90	195.28	73.94

The scalability of the algorithms has been initially investigated by varying the value of *minsupp* in a range that can reasonably cover real cases, i.e. $[0.05, 0.35]$: with lower values, the number of frequent patterns becomes too large, hampering the comprehension of the results, and with higher values almost no pattern is present.

Figure 2 shows the recorded runtimes (in milliseconds) for increasing values of *minsupp* for all the algorithms evaluated over all the three datasets. Each sub-figure refers to one specific dataset, as indicated in the captions: a sub-chart shows the runtime of different algorithms over one specific target dataset. For the sake of clarity, a logarithmic scale has been used for the vertical axis, with the same range of values, to make it easier telling apart the different curves in the charts. It can be noted that, as expected, in general the higher the *minsupp* value, the lower the number of retrieved frequent

*https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data

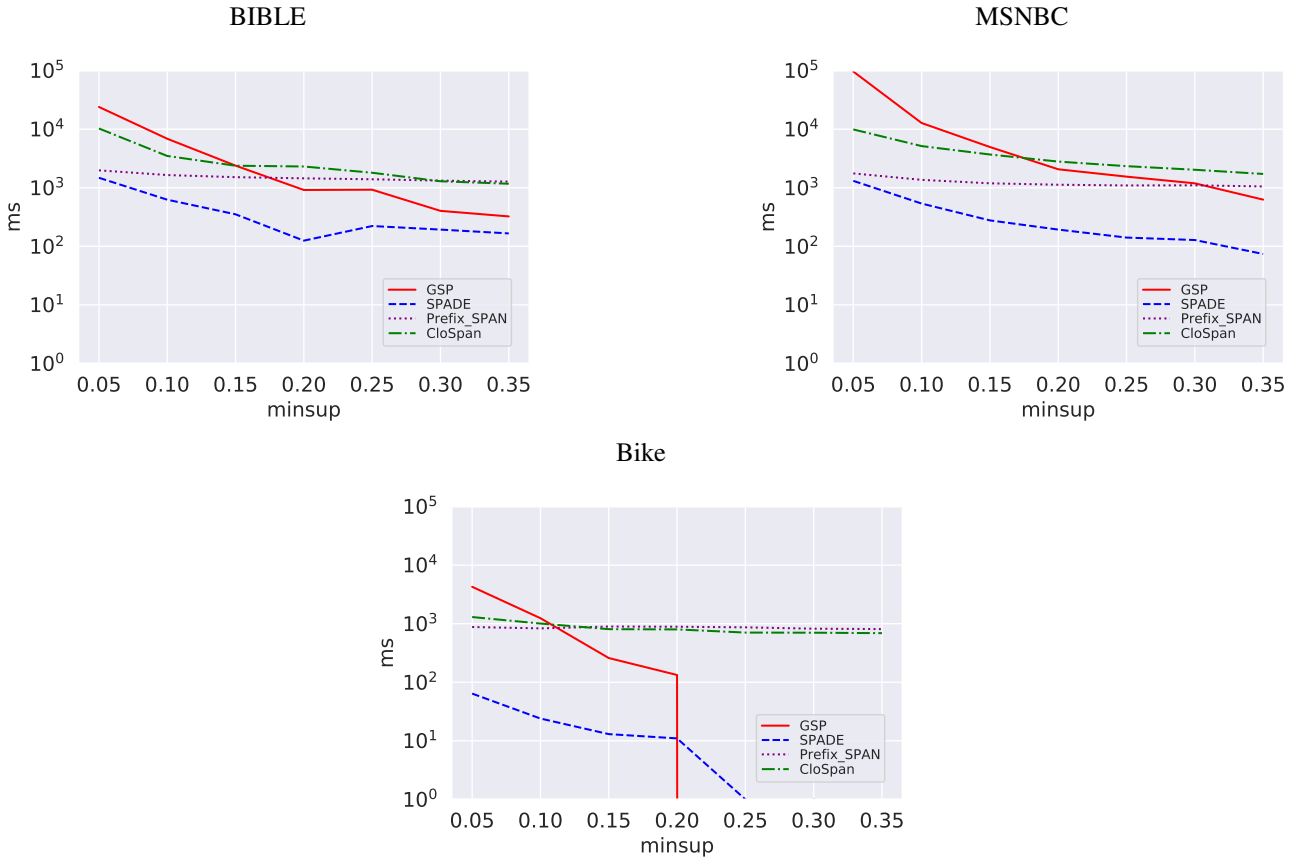

Figure 2. Runtimes obtained over the reference datasets by the target algorithms.

sequence patterns, and the sooner the algorithms complete their execution. The variability across datasets is higher for GSP and SPADE, whereas the runtimes of PrefixSpan and Clo-Span assume similar values. Even if each dataset stresses the various algorithms in different ways, we can see that, for low values of *minsupp*, GSP performs poorly in any case, whereas SPADE behaves quite well. Moreover, performance is typically affected by the number of sequences, and the sequence length has a less important impact: in fact, runtimes for the Bike dataset are always shorter for all the algorithms.

The size of the target dataset is one of the most influential parameters for the behavior of an SPM algorithm, so it is important to assess scalability also with respect to the dataset size. To this aim, we decided to run the algorithms with increasing fractions of the MSNBC dataset, looking at the recorded trend for both runtime and memory footprint, as already done in the previous experiments. Figure 3 shows the memory footprint (in Mb) with respect to the dataset size. Although relative differences can be noted on the curves, both for runtimes and for memory footprint, the general trend with respect to the dataset size is similar for all the algorithms.

4. Techniques related to Sequential Pattern Mining

Many different problems that are strongly related to the field of SPM attracted the interest of the research community in the last years, giving rise to several distinct investigation lines. For example,

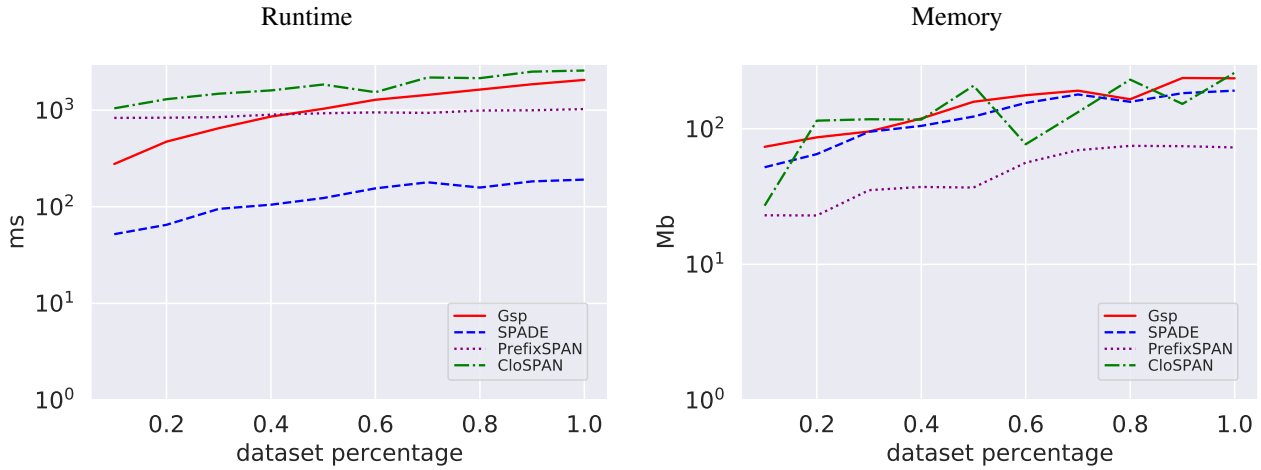

Figure 3. Scalability assessment for all the algorithms with increasing fractions of MSNBC dataset.

Periodic Pattern Mining consists of finding patterns that appear frequently and periodically in a single long sequence. The pattern periodicity is measured based on its period lengths, [53, 83]. A generalization regarding the ordering of events leads to the adoption of *graph databases*, suited to model itemsets along with general relationships across them. In this case, the SPM problem can be re-cast to *Sub-graph Mining*: It consists of finding *frequent sub-graphs* in a database of graphs or in one single large graph [110], [51].

Some specific problems, closely related to SPM, have recently shown a growing importance. For this reason, in the following we take a look at the most notable of them, and at solutions proposed for them.

4.1. Weighted and High-Utility SPM

So far, we have discussed about events and objects in sequences, assuming that all of them would be equally important for the sake of the addressed application, but in real settings this is not always the case. Thus, we can reformulate the SPM problem by explicitly considering different degrees of importance: Weighted SPM and, subsequently, High-Utility Mining are the two most popular approaches of this kind in the literature. In the case of Weighted SPM, each item in the sequence database is typically assigned a weight in the range $[0, 1]$, denoting its relevance. Of course, it is necessary to adapt classical SPM algorithms to handle weights as well. An example is *WSPAN* [117], whose authors propose to use a weight range and prune weighted infrequent sequential patterns, thus generating fewer sequential patterns with higher overall weights.

High-Utility SPM can be considered an extension of Weighted SPM. In this case, however, rather than using a weight for each object in the sequence database, it relies on the fact that each object/item may appear zero, one or more times within a transaction. Thus, each item is weighted with respect to its relative importance within the transaction or dataset. In addition to the minimum support threshold required for the identification of frequent sequential patterns, a *utility threshold* is taken into account. Thus, to be considered frequent, a specific pattern must have a support greater than or

equal to *minsupp*, and it must also exceed a utility threshold, which could be relative either to each item or to the total number of items in a transaction. A widely used High-Utility Pattern Mining algorithm is *USPAN* [114], which exploits a lexicographic quantitative sequence tree to represent the sequences, along with a set of concatenation mechanisms to identify High-Utility ones. Notably, recent research has investigated solutions for High-Utility SPM for Big Data, devising algorithms that leverage the MapReduce distributed programming model [59], and attention has been devoted also to efficient mining of High-Utility Sequences with constraints [94].

4.2. Multi-dimensional SPM

Another problem studied in the literature is that of Multi-dimensional SPM, introduced by [78]. According to the authors, traditional SPM approaches allow the identification of global regularities within the database, but not the ability to focus on specific problems. For example, given a pattern identified as frequent in a sequence database containing customer transactions, it may be globally frequent. However, when considering a specific category of customers (for example, "over 55"), the same conclusion may not be true. To solve this problem, sequence databases are enriched with annotations concerning characteristic traits of the sequences. Frequent sequences are identified also along specific dimensions. Certain sequences may thus be frequent for some values of those dimensions but infrequent for others [91]. At first sight, the idea may appear similar to that of quantitative sequences, discussed in Section 3.4. However, while for quantitative sequences the goal is to model quantities of the actual data, such as how many items in a transaction, in this case the aim is to model both the data with respect to some kind of metadata that is complementary to the data of the transaction. SeqDIM [78] is a popular example of Multi-dimensional SPM algorithm. It can be considered as a meta-algorithm, as it requires a sequential pattern mining algorithm for discovering sequential patterns and an itemset mining algorithm to deal with the dimensions.

4.3. Stream SPM

Often a large amount of information can be available in real time in the form of data streams, and knowledge should be extracted from them in real time as well: techniques under the name of "Stream SPM" have been proposed for this purpose, recently gaining recognition in the literature. The proposed algorithms are typically extensions of incremental algorithms, such as the aforementioned SPADE. The key constraint that Stream SPM algorithms must face is, obviously, the fact that they must be able to identify sequential patterns even if the entire sequence cannot be read more than once, since it is a stream of data in continuous evolution. Some examples of popular algorithms for Stream SPM are eISeq [12], IncSPAM [46], SPEED [81], and Seqstream [13].

4.4. Uncertain and fuzzy SPM

Data collected in a real setting can frequently be affected by uncertainties and noise, and thus their analysis must take into account such unwanted characteristics. Uncertainty in SPM can occur in three different places: the assignment of an event to the proper sequence (or "source"), the event itself, and the event timestamp [71]. Temporal uncertainty can be due to various reasons. For example, conflicting or missing event timestamps, network latency, granularity mismatches, synchronization problems, device precision limitations, data aggregation. A significant specific case refers to temporal

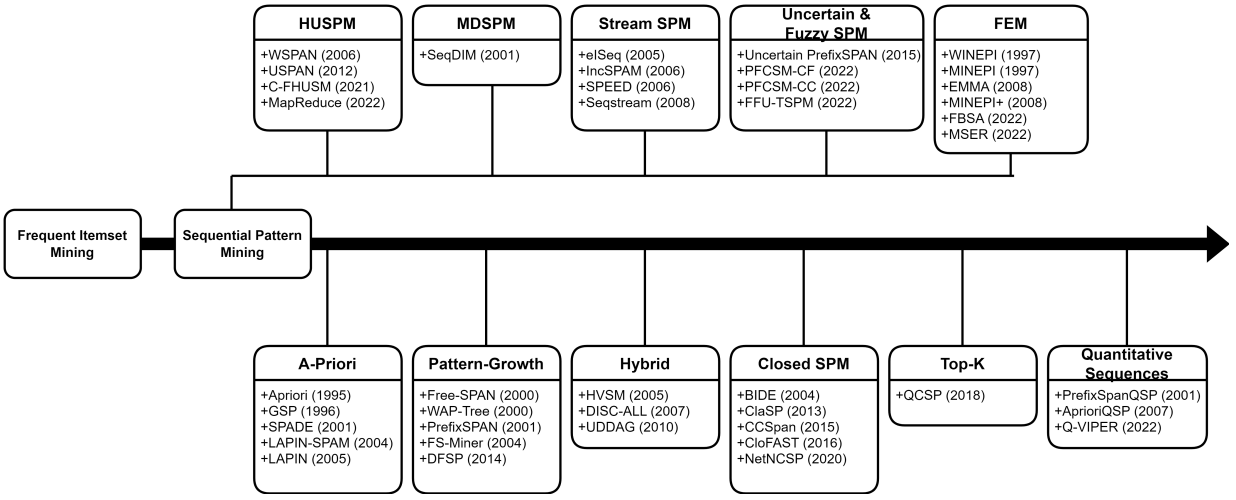

Figure 4. Problems and relative algorithms discussed in this work.

placement of events [40]. To overcome the difficulties deriving from inexact values, or noisy datasets, specific approaches like Uncertain SPM and Fuzzy SPM have been developed. Typically, probabilistic techniques and fuzzy logic solutions are exploited. For example, it was proposed an adapted version of PrefixSpan [71] that exploits dynamic programming to compute the expected support of a sequential pattern.

Efficiency and scalability are challenging issues that emerge in SPM applied to uncertain databases. An approach proposed to tackle such issues is based on dynamic programming to mine probabilistic frequent sequential patterns over the distributed platform Spark [39]. For addressing the problem of uncertain data that affects also the High-Utility sequential patterns, a high average-utility sequential pattern mining framework [58] has been proposed for discovering the set of potentially high average-utility sequential patterns by considering the sequence size. A drawback of several recent works that attempt to mine frequent uncertain sequential patterns is their low efficiency in reducing the number of generated false-positive patterns in their mining process. To address this issue, it has been shown that the exploitation of a hierarchical index structure may lead to promising results [85].

The mining of closed sequences in uncertain data has proved to be very challenging. The PFCSM-CF and PFCSM-CC algorithms [115] have been designed to reduce the search space and simplify the candidate sequence database, with valuable computational benefits.

The exploitation of fuzzy set theory has shown to be useful in presence of uncertainties in SPM. For example, in [118] the authors have proposed an algorithm based on a sliding window constraint that allows an element to be considered as part of different transactions within a user specified window. More recently, fuzziness has been used also to deal with High-Utility Fuzzy Sequential Patterns [84].

For the sake of clarity, all the problems and the corresponding algorithms discussed in this survey are shown in Figure 4.

5. Frequent Episode Mining

In recent years, another important related pattern mining problem that received particular attention is *Frequent Episode Mining*, and promising solutions have been put forward. The discovery of sub-sequences whose instances occur frequently along a single sequence, each one according to a specific temporal order, is a very relevant problem in many application domains, especially in the context of system log analysis [122]. In fact, while several applications in SPM deal with sequential data that is typically coming from many small sequences (for example, a database of customer transactions for a company), a notable exception is identifiable in domains where the target phenomenon is described by a single long sequence of events, and for which patterns of interest are small sub-sequences. Examples of applications in such domain are the monitoring of network traffic for detecting attacks, the monitoring of telecommunication alarm signals, or the analysis of usage data for recurring patterns.

In this setting, an event is always associated with a timestamp, and it may contain one or more items out of a given set. The frequent sub-sequences of events that we are interested in are usually called *episodes*.

The Frequent Episode Mining (FEM) problem was first introduced by Mannila et al. [66]. In their work, authors define an episode as a collection of events that occur within time intervals of a given size in a given partial order [66]. Thus, given a sequence of events, each marked by a specific timestamp, and a predefined time window, the FEM objective is to identify *episodes* of events, or sub-sequences, that occur frequently together at a specific time and with a specific partial ordering.

A formal definition is given in [34]. In the most general case, some events can occur concurrently, i.e. may have the same timestamp; to emphasize this aspect, sequences with possible concurrent events have been called *complex sequences*. Consider a finite set of items $I = \{i_1, i_2, ..., i_m\}$.

A complex sequence S of events, each with one or more items out of I, is represented by a temporally ordered list of tuples of the type (SE_{t_i}, t_i), where SE_{t_i} is the event occurring at timestamp t_i. An episode is thus a non-empty, totally-ordered set of events of the form $\langle E_1, E_2, ..., E_p \rangle$, where E_i is a subset of I and E_i appears before E_j for all integers i and j, with $i < j$, in the interval $[1, p]$. The purpose of FEM is therefore to identify all frequent episodes, i.e. those with high support, in a sequence S. The occurrence of an episode is given by a time interval $[t_s, t_e]$ where t_s and t_e are the start and end timestamps for the episode, respectively. The support of an episode is clearly given by the number of its occurrences within the sequence. In the literature, several methods have been proposed to calculate the support. Older works proposed to calculate it through the minimum number of occurrences of an episode [66], while more recent works typically exploit the frequency of the episode head, since it has been experimentally proven to be the best measure of support for this problem [50]. Given a user-defined *winlen* length window, and a minimum *minsupp* support threshold, the goal is to identify episodes that occur at least *minsupp* times within a *winlen* window.

Several algorithms have been proposed in the literature to address the FEM problem. The authors in [66] first addressed the problem, and proposed the WINEPI and MINEPI algorithms. They follow largely the same procedure, but differ in the calculation of frequency and support. In WINEPI, an episode is only frequent when all its sub-episodes are also frequent. On the other hand, MINEPI exploits the minimum occurrences mentioned above. Thus, one of its advantages over WINEPI is that rules are produced with two specific time limits. The authors claim that MINEPI can outperform WINEPI in the last iterations of the algorithm. In [50] EMMA and MINEPI+ were proposed. Both

improve WINEPI and MINEPI by using head frequency counting to calculate sequence support. Algorithms focused on High-Utility episode mining have also been proposed [33, 104]. In this case, the goal of the algorithms is to identify High-Utility episodes in complex event sequences such as transaction databases. Recently, the problem of mining serial episode rules from complex event sequences emerged. The Forward and Backward Search Algorithm (FBSA) was developed to detect minimal occurrences of frequent peak episodes and to eliminate the frequent sequence scans and redundant event sets [79]. A new technique called Mining Serial Episode Rules (MSER) was also proposed and it is based on the correlation between episodes and the generation of parameter selection where the occurrence time of an event is specified in the consequent. Efforts have also been made on mining Frequent Serial Episodes over data streams [42].

Given that episode mining is an NP-complete problem, and that the *minsupp* parameter has a rather high influence on the quality and quantity of the results (i.e. too many episodes or uninteresting episodes), algorithms have been proposed for mining frequent top-k episodes in a sequence, either focusing on the standard setting [34] or for High-Utility mining [82]. A scalable, distributed framework to support FEM on event sequences either very long or with masses of simultaneous events was proposed in [5]. It is an event-centered and hierarchy-aware partitioning strategy aimed at allocating events from different levels of the hierarchy into local processes, extended also to support maximal and closed episode mining in the context of event hierarchy. In addition, the authors of this framework also presented an algorithm to improve the local mining performance.

Finally, several applications of the FEM algorithms have been developed in the last few years. For example, FEM algorithms have been used in building a prediction model aimed at solving the resource provisioning problem in the cloud environment [3]. Another kind of application is the automation of sequences of repetitive actions performed by human operators in interacting with software applications. Robot applications can automatise sequences of logs to prevent work perceived as alienating and boring by employees. A two-step approach to mine sequences of actions to be automated from log data produced by interactions of a human operator with specific software application has been proposed very recently in [21].

6. Available software for Sequential Pattern Mining

The field of sequential pattern mining is clearly very oriented to the development of algorithmic solutions focused on pushing the state-of-the-art particularly towards efficiency as dataset sizes increase. As it regards the actual implementations of algorithms presented in the literature, often authors delve into thorough descriptions of their proposals, possibly assessing the relative performances to compare them with state-of-the-art approaches, but in the context of the paper they rarely make available open-source code to the scientific community. This is especially the case for many classical algorithms that have been devised several years ago. The reason of the lack of publicly available code may be due to commercial restrictions, or to the fact that the algorithm itself has been implemented in specific architectures that are harder to replicate in a completely open and self-contained framework such as for distributed computing. Clearly, this has led over the years to the creation of resources and small third-party libraries specifically designed for the implementation of certain algorithms in the literature [31]. On the one hand, a number of classical algorithms such as

Apriori,[†] FPGrowth,[‡] and PrefixSpan [§] to name a few have been coded by several users in open source communities such as GitHub. On the other hand, open-source libraries specifically designed to address SPM techniques at large and incorporate a wide array of implemented algorithms from the literature, are more scarce. Nevertheless, a few notable exceptions can be identified.

It is worth underlining that many open-source or at the very least freeware libraries for data mining and machine learning, such as for example the popular Weka [102] and KNIME [9], include implementations of basic standard algorithms for SPM. Notably, also the MLlib library of the Spark framework supports some basic algorithms, such as FP-Growth and Prefix-Span.[¶] Additionaly, a widely popular alternative has been proposed by means of SPMF (Sequential Pattern Mining Framework).[‖] As the name suggests, it includes the implementation of a large number of algorithms for SPM and related problems, such as high-utility pattern mining, episode mining, and stream mining among others [31]. Written in Java, it includes a large part of the algorithms described in the previous sections. The framework is being actively maintained at the time of writing.

In recent years, Python has become one of the most popular languages in the data science community [10]. Anyway, currently the number of open, comprehensive Python implementations of SPM techniques is very limited, and popular data mining toolkits like SciKit-Learn [73] have very limited coverage of SPM-related algorithms. This might be due to the fact that SPM techniques usually require efficient systems and languages, and Python is not considered one of them, even if the situation is rapidly changing. In this perspective, it is worth mentioning also the recently released PAMI[**] framework. It is an open-source Python library for SPM and related problems. The main advantage of PAMI over other similar frameworks is, according to the words of the authors, that the library can better "satisfy the information needs of the data scientists who perform analytics on Big Data mostly using Python language." Even if SPFM can be considered definitely more extensive in terms of available algorithms, PAMI includes several traditional SPM algorithms, as well as distributed and GPU-accelerated ones, thus enabling a wider array of possibilities especially for big data analytics. Moreover, other comprehensive Python resources are emerging: a notable example is in the field of constraint-based Sequence Mining algorithms, and it is represented by the Seq2Pat library [101], which is freely available on the web: it focuses on practical issues like scalability, explainability, rapid experimentation, and reusability.

7. Future directions

The research questions posed by the literature on Sequential Pattern Mining and related problems have been tackled in several different ways over the last three decades. Despite the wide variety of application contexts, the basic formal model used for SPM (as summarized in Section 2) has been able to accommodate most of the specific needs. Even if the problem could not be considered a novel one, there are still many open challenges and potential applications to explore, thus making it an intriguing research topic also nowadays.

[†] https://github.com/tommyod/Efficient-Apriori
[‡] https://github.com/chonyy/fpgrowth_py
[§] https://github.com/chuanconggao/PrefixSpan-py
[¶] https://spark.apache.org/docs/1.5.0/mllib-frequent-pattern-mining.html
[‖] https://www.philippe-fournier-viger.com/spmf/
[**] https://github.com/udayRage/PAMI

Some past reviews on SPM [32, 45, 70, 123] identified several possible future developments in the field and now we can check what it has been accomplished, what it must still be done, and what new perspectives we are facing today. One key aspect pointed out in different ways by almost all earlier reviews is the need to address, beyond the computational efficiency, the practical utility and interpretability of the results, whose possible huge size practically hampers their effective usage. As discussed in Section 3.4, this aspect has been addressed by considering different possible types of constraints, and significant results have been obtained, as witnessed for example by recent proposals for extracting only the best k patterns [34]. Anyway, we must admit that a semantic approach to the identification of the most meaningful results for the end user is still missing, and can thus be considered a challenge for future investigations. Regarding the semantic aspects of SPM, some questions have been formulated: "what is the meaning of the pattern? What are the synonym patterns? And what are the typical transactions that this pattern resides? Why a certain pattern is frequent?" [45]. Certainly, any attempt for an answer still requires extensive research efforts, and the integration of ontologies has been proposed as a future development as well [70].

Ideas and techniques originally developed for SPM can provide significant results in related application areas, as in the case of time series analysis, where in a recent study they have shown to be able to effectively point out temporal behaviors or trends [109]. Hopefully, many other examples of such cross-contamination in other fields could be expected.

The opportunity to use some forms of temporal logic has been advocated for a better specification of the desired patterns, or for a more precise identification of rules [70], but so far no proposal has emerged as a general agreed specification framework for temporal aspects in SPM. Instead, application-specific requirements led to the development of ad-hoc solutions: we suppose that the variety of needs in disparate applications could be an obstacle in pursuing this possible research line.

The growing availability of data from different sources and domains is definitely an incentive to provide more efficient algorithms that leverage modern computing hardware, such as distributed computing and GPU acceleration [32]. We have shown in the previous Sections how a number of approaches already tackled the issues of parallelization, especially in leveraging MapReduce-like algorithms and frameworks such as Spark. More recently, approaches have also been proposed focusing on cloud [4] and fog [11] computing environments for leveraging Big Data and IoT-enabled devices to perform sequential pattern mining. Given the fact that IoT-enabled devices can serve both as computational power and for data collection, an interesting area of research is that of Stream SPM in this specific context.

GPUs are likely among the most promising computing platforms for delivering further performance in SPM techniques, also considering the fact that GPU acceleration has drastically increased the capabilities of deep learning models in the last few years. Some efforts have been made also in this direction [48, 72], showing how leveraging GPUs can decrease the computation time by orders of magnitude with respect to traditional computing hardware.

The inherent peculiarities of large amounts of available data represent an interesting challenge along two different perspectives. First, in terms of strategies for dealing with the increasing complexity in represented information and related data structures [32]. Although most of the literature focused on specific benchmark datasets and application domains such as transactional databases, this growing complexity could be the driver for a large number of researches aimed at modelling diverse characteristics of the data in various contexts. Second, the spectrum of potential applications for

sequential pattern mining algorithms could be broaden. For example, in the bioinformatics domain several algorithms have been studied for mining patterns in genetic sequences [65, 87]. Other potential fields of application are: usage mining, such as the analysis of user interactions with applications or web pages to understand their behavioural patterns, and how end users interact with a specific platform, also in order to suggest automation strategies [18]; telecommunications, such as mining of user movement patterns, and next location prediction; intrusion detection, especially for client-server applications, based on activity logs, that can be exploited to mine potentially suspicious patterns. In general, we can argue that any application dealing with inherently sequential data could be explored also in the context of SPM.

An interesting perspective in this regard can also be obtained by looking at trends in other fields of machine learning. In recent years, Attention-based models [99] such as Transformers have imposed themselves as the de-facto standard for many machine learning tasks, including Computer Vision and Natural Language Processing. An interesting aspect is that, at their core, such architectures aim to model sequences (e.g., of words, patches of images, genome). It would undoubtedly be interesting understanding how they relate to each other in terms of discovering and predicting new patterns in data, and how these two different paradigms for learning from sequences could influence each other.

8. Conclusions

The main goal of this paper was to provide an overview of the research field on Sequential Pattern Mining (SPM) approaches and algorithms, as well as related problems. In order to do so, we adopted two categorization of the approaches proposed in the literature.

First, we categorized them through the specific problem of SPM, considering differences in behaviour and/or utilized constraints. Our main goal was to provide an overview of the research field, adopting a categorization in terms of the approach they pursue, and presenting the most significant and popular algorithms for each category. The description has been complemented with an experimental analysis for getting insights into the performances of some of the most well-known algorithms on benchmark datasets for pattern mining. From this first categorization, as well as from the performed experiments, we can conclude that there are not clear differences between the various approaches, both in terms of computational cost/efficiency and results. As a general rule of thumb, considering that most of the algorithms build upon each other, practitioners and researcher looking to provide improved solutions may first consider newer and hybrid implementations (for example, top-k SPM) as the state-of-the-art. Further, due to the fact that in target applications sequential data are generally rather large, parallelized versions of SPM algorithms may be a good choice, provided the availability of computational resources.

Second, we categorized different approaches that are strongly related to the problem of mining sequential patterns. We included Weighted and High-Utility SPM (HUSPM), Multi-Dimensional SPM, Stream SPM, Fuzzy SPM, and Frequent Episode Mining (FEM). In this regard, it is clear that the choice of one approach over others is mostly related to (i) the available data and (ii) the task at hand. For example, HUSPM may prove helpful when dealing with sequential data where certain events, or items, are considered to be of higher importance or are associated with specific benefit. Multi-Dimensional SPM may be the best choice when dealing with multi-faceted data that can be explored along different dimensions with different results, such as customer transactions with

associated demographic information. On the other hand, Stream and Fuzzy SPM are particularly useful when the data is either continuously updating or is missing information. Finally, FEM may be the best fit for datasets that include long sequences that have to be explored also horizontally, or when the data is actually a single, long, sequence, from which we aim to mine recurring patterns, as for example in the case of log analysis.

We provided also an account of several practical options for employing SPM algorithms in custom applications, in particular by looking at open-source code currently available for pattern mining. Finally, by revising expectations expressed in previous surveys on the SPM subject, we proposed some potentially interesting future research directions.

Even if SPM algorithms have been a staple of the research on data mining for almost the last three decades, upon a survey of related contributions in the recent literature we can state that this is still an active research field with a vast number of unexplored options from the application standpoint, and room for improvement for what concerns the efficiency and effectiveness of algorithms for pattern mining under various lights.

Acknowledgments

This work was partially supported by Tuscany Region in the context of the project "AUTOMIA" in the framework of regional program "POR FESR Toscana 2014-2020".

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. R. Agrawal, R. Srikant, Mining sequential patterns, *Proceedings of the eleventh international conference on data engineering*, (1995), 3–14.
2. R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, (1993), 207–216. https://dx.doi.org/10.1145/170035.170072
3. M. Amiri, L. Mohammad-Khanli, R. Mirandola, An online learning model based on episode mining for workload prediction in cloud, *Future Generation Computer Systems*, **87** (2018), 83–101. https://doi.org/10.1016/j.future.2018.04.044
4. M. Amiri, L. Mohammad-Khanli, R. Mirandola, A sequential pattern mining model for application workload prediction in cloud environment, *J. Netw. Comput. Appl.*, **105** (2018), 21–62. https://dx.doi.org/10.1016/j.jnca.2017.12.015
5. X. Ao, H. Shi, J. Wang, L. Zuo, H. Li, Q. He, Large-scale frequent episode mining from complex event sequences with hierarchies, *ACM T. Intel. Syst. Tec. (TIST)*, **10** (2019), 1–26. https://doi.org/10.1145/3326163
6. J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, (2002), 429–435. https://dx.doi.org/10.1145/775047.775109

7. M. Barsacchi, A. Bechini, F. Marcelloni, Implicitly distributed fuzzy random forests, *Proc. of the 36th Annual ACM Symposium on Applied Computing*, (2021), 392–399. https://dx.doi.org/10.1145/3412841.3442082
8. K. Beedkar, R. Gemulla, Lash: Large-scale sequence mining with hierarchies, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, (2015), 491–503. https://dx.doi.org/10.1145/2723372.2723724
9. M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, et al., KNIME: The Konstanz Information Miner. In: Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007), Springer. https://doi.org/10.1007/978-3-540-78246-9_38
10. S. Biswas, M. Wardat, H. Rajan, (2022) The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large, *Proc. of the 44th International Conference on Software Engineering*, (2022), 2091–2103, https://dx.doi.org/10.1145/3510003.3510057
11. P. Braun, A. Cuzzocrea, C. K. Leung, A. G. M. Pazdor, J. Souza, S. K. Tanbeer, Pattern mining from big iot data with fog computing: Models, issues, and research perspectives, *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, (2019), 584–591, https://dx.doi.org/10.1109/CCGRID.2019.00075
12. J. H. Chang, W. S. Lee, Efficient mining method for retrieving sequential patterns over online data streams, *J. Inf. Sci.*, **31** (2005), 420–432. https://dx.doi.org/10.1177/0165551505055405
13. L. Chang, T. Wang, D. Yang, H. Luan, Seqstream: Mining closed sequential patterns over stream sliding windows, *2008 Eighth IEEE International Conference on Data Mining*, (2008), 83–92. https://dx.doi.org/10.1109/ICDM.2008.36
14. C. C. Chen, C. Y. Tseng, M. S. Chen, Highly scalable sequential pattern mining based on MapReduce model on the cloud, *2013 IEEE International Congress on Big Data*, (2013), 310–317. https://dx.doi.org/10.1109/bigdata.congress.2013.48
15. C. C. Chen, H. H. Shuai, M. S. Chen, Distributed and scalable sequential pattern mining through stream processing, *Knowl. Inf. Syst.*, **53** (2017), 365–390. https://dx.doi.org/10.1007/s10115-017-1037-1
16. J. Chen, An updown directed acyclic graph approach for sequential pattern mining, *IEEE T. Knowl. Data En.*, **22** (2009), 913–928. https://dx.doi.org/10.1109/TKDE.2009.135
17. D. Y. Chiu, Y. H. Wu, A. L. Chen, An efficient algorithm for mining frequent sequences by a new strategy without support counting, *Proceedings. 20th International Conference on Data Engineering*, (2004), 375–386, https://dx.doi.org/10.1109/ICDE.2004.1320012
18. D. Choi, H. R'bigui, C. Cho, Candidate digital tasks selection methodology for automation with robotic process automation, *Sustainability*, **13** (2021), 8980. https://dx.doi.org/10.3390/su13168980
19. S. Cong, J. Han, D. Padua, Parallel mining of closed sequential patterns, *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, (2005), 562–567. https://dx.doi.org/10.1145/1081870.1081937

20. T. J. Czubryt, C. K. Leung, A. G. M. Pazdor, Q-VIPER: Quantitative vertical bitwise algorithm to mine frequent patterns. In: Wrembel R, Gamper J, Kotsis G, Tjoa AM, Khalil I (eds) *Big Data Analytics and Knowledge Discovery, Springer International Publishing*, (2022), 219–233. https://dx.doi.org/10.1007/978-3-031-12670-3_19
21. P. Dell'Oglio, A. Bondielli, A. Bechini, F. Marcelloni, Leveraging sequence mining for robot process automation. In: Abraham A, Pllana S, Casalino G, Ma K, Bajaj A (eds) *Intelligent Systems Design and Applications - 22nd International Conference on Intelligent Systems Design and Applications (ISDA 2022) held December 12-14*, 2022, Springer Nature Switzerland AG, in press.
22. A. Demiriz, webSPADE: a parallel sequence mining algorithm to analyze web log data, *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, (2002), 755–758. IEEE. https://dx.doi.org/10.1109/icdm.2002.1184046
23. M. El-Sayed, C. Ruiz, E. Rundensteiner, Fs-miner: Efficient and incremental mining of frequent sequence patterns in web logs, *Proc. of the International Workshop on Web Information and Data Management*, (2004), 128–135. https://dx.doi.org/10.1145/1031453.1031477
24. C. I. Ezeife, Y. Lu, Y. Liu, Plwap sequential mining: open source code, *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, (2005), 26–35. https://dx.doi.org/10.1145/1133905.1133910
25. Y. Fan, Y. Ye, L. Chen, Malicious sequential pattern mining for automatic malware detection, *Expert Systems with Applications*, **52** (2016), 16–25. https://dx.doi.org/10.1016/j.eswa.2016.01.002
26. X. Fei, S. Zheng, Yan Lj, Fan C (2016) A improved sequential pattern mining algorithm based on PrefixSpan, *2016 World Automation Congress (WAC)*, (2016), 1–4. IEEE. https://dx.doi.org/10.1109/wac.2016.7583059
27. L. Feremans, B. Cule, B. Goethals, Mining top-k quantile-based cohesive sequential patterns, *Proceedings of the 2018 SIAM international conference on data mining*, (2018), 90–98.
28. P. Fournier-Viger, C. W. Wu, V. S. Tseng, Mining maximal sequential patterns without candidate maintenance, *International Conference on Advanced Data Mining and Applications*, (2013), 169–180. Springer Berlin Heidelberg. https://dx.doi.org/10.1007/978-3-642-53914-5_15
29. P. Fournier-Viger, T. Gueniche, S. Zida, V. S. Tseng, Erminer: sequential rule mining using equivalence classes, *International Symposium on Intelligent Data Analysis*, (2014), 108–119. https://doi.org/10.1007/978-3-319-12571-8_10
30. P. Fournier-Viger, C. W. Wu, A. Gomariz, V. S. Tseng, VMSP: Efficient vertical mining of maximal sequential patterns, *Canadian conference on artificial intelligence*, (2014), 83–94. Springer International Publishing. https://dx.doi.org/10.1007/978-3-319-06483-3_8
31. P. Fournier-Viger, J. C. W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, et al., The SPMF open-source data mining library version 2, *Joint European conference on machine learning and knowledge discovery in databases*, (2016), 36–40. Springer. https://dx.doi.org/10.1007/978-3-319-46131-1_8
32. P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, R. Thomas, A survey of sequential pattern mining, *Data Science and Pattern Recognition*, **1** (2017),54–77.

33. P. Fournier-Viger, P. Yang, J. C. W. Lin, U. Yun, Hue-span: Fast high utility episode mining, *International Conference on Advanced Data Mining and Applications*, (2019), 169–184. https://dx.doi.org/10.1007/978-3-030-35231-8_12
34. P. Fournier-Viger, Y. Yang, P. Yang, J. C. W. Lin, U. Yun, TKE: Mining top-k frequent episodes, *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, (2020), 832–845. https://dx.doi.org/10.1007/978-3-030-55789-8_71
35. J. Fowkes, C. Sutton, A subsequence interleaving model for sequential pattern mining, *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, (2016), 835–844. https://dx.doi.org/10.1145/2939672.2939787
36. F. Fumarola, P. F. Lanotte, M. Ceci, D. Malerba, CloFAST: closed sequential pattern mining using sparse and vertical id-lists, *Knowl. Inf. Syst.*, **48** (2016), 429–463. https://dx.doi.org/10.1007/s10115-015-0884-x
37. W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, P. S. Yu, A survey of parallel sequential pattern mining, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **13** (2019), 1–34. https://dx.doi.org/10.1145/3314107
38. M. Garofalakis, R. Rastogi, K. Shim, Mining sequential patterns with regular expression constraints, *IEEE T. Knowl. Data En.*, **14** (2002), 530–552. https://dx.doi.org/10.1109/TKDE.2002.1000341
39. J. Ge, Y. Xia, Distributed sequential pattern mining in large scale uncertain databases, *Pacific-Asia conference on knowledge discovery and data mining*, (2016), 17–29. Springer. https://dx.doi.org/10.1007/978-3-319-31750-2_2
40. J. Ge, Y. Xia, J. Wang, C. H. Nadungodage, S. Prabhakar, Sequential pattern mining in databases with temporal uncertainty, *Knowl. Inf. Syst.*, **51** (2017), 821–850. https://dx.doi.org/10.1007/s10115-016-0977-1
41. A. Gomariz, M. Campos, R. Marin, B. Goethals, Clasp: An efficient algorithm for mining frequent closed sequences, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (2013), 50–61. Springer. https://dx.doi.org/10.1007/978-3-642-37453-1_5
42. T. Guyet, W. Zhang, A. Bifet, Incremental mining of frequent serial episodes considering multiple occurrence, *Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part I.*, Cham: Springer International Publishing. https://dx.doi.org/10.48550/ARXIV.2201.11650
43. J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. Hsu, Freespan: Frequent pattern-projected sequential pattern mining, *Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2000), 355–359. https://dx.doi.org/10.1145/347090.347167
44. J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, et al., PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth, *proceedings of the 17th international conference on data engineering*, (2001), 215–224. https://dx.doi.org/10.1109/icde.2001.914830
45. J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: Current status and future directions, *Data Min. Knowl. Disc.*, **15** (2007), 55–86. https://dx.doi.org/10.1007/s10618-006-0059-1

46. C. C. Ho, H. F. Li, F. F. Kuo, S. Y. Lee, Incremental mining of sequential patterns over a stream sliding window, *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)*, (2006), 677–681. IEEE. https://dx.doi.org/10.1109/ICDMW.2006.98
47. A. Hosseininasab, W. J. van Hoeve, A. A. Cire, Constraint-based sequential pattern mining with decision diagrams, *Proc of the AAAI Conference on Artificial Intelligence*, **33** (2019), 1495–1502. https://dx.doi.org/10.1609/aaai.v33i01.33011495
48. Y. H. Hsieh, C. C. Chen, H. H. Shuai, M. S. Chen, Highly parallel sequential pattern mining on a heterogeneous platform, *2018 IEEE International Conference on Data Mining (ICDM)*, (2018), 1037–1042. https://dx.doi.org/10.1109/ICDM.2018.00131
49. J. W. Huang, S. C. Lin, M. S. Chen, DPSP: Distributed progressive sequential pattern mining on the cloud, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (2010), 27–34. https://dx.doi.org/10.1007/978-3-642-13672-6_3
50. K. Y. Huang, C. H. Chang, Efficient mining of frequent episodes from complex sequences, *Inform. Syst.*, **33** (2008), 96–114. https://dx.doi.org/10.1016/j.is.2007.07.003
51. C. Jiang, F. Coenen, M. Zito, A survey of frequent subgraph mining algorithms, *The Knowledge Engineering Review*, **28** (2013), 75–105.
52. C. Kim, J. H. Lim, R. T. Ng, K. Shim, SQUIRE: Sequential pattern mining with quantities, *J. Syst. Software*, **80** (2007), 1726–1745. https://dx.doi.org/10.1016/j.jss.2006.12.562
53. R. U. Kiran, M. Kitsuregawa, P. K. Reddy, Efficient discovery of periodic-frequent patterns in very large databases, *J. Syst. Software*, **112** (2016), 110–121.
54. B. Le, H. Duong, T. Truong, P. Fournier-Viger, Fclosm, fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy, *Knowl. Inf. Syst.*, **53** (2017), 71–107. https://dx.doi.org/10.1007/s10115-017-1032-6
55. P. Lenca, B. Vaillant, P. Meyer, S. Lallich, Association rule interestingness measures: Experimental and theoretical studies, *Quality Measures in Data Mining*, (2007), 51–76.
56. Y. Liang, S. Wu, Sequence-growth: A scalable and effective frequent itemset mining algorithm for big data based on MapReduce framework, *2015 IEEE International Congress on Big Data*, (2015), 393–400. IEEE. https://dx.doi.org/10.1109/BigDataCongress.2015.65
57. V. C. C. Liao, M. S. Chen, DFSP: a Depth-First SPelling algorithm for sequential pattern mining of biological sequences, *Knowl. Inf. Syst.*, **38** (2014), 623–639. https://dx.doi.org/10.1007/s10115-012-0602-x
58. J. C. W. Lin, T. Li, M. Pirouz, J. Zhang, P. Fournier-Viger, High average-utility sequential pattern mining based on uncertain databases, *Knowl. Inf. Syst.*, **62** (2020), 1199–1228. https://doi.org/10.1007/s10115-019-01385-8
59. J. C. W. Lin, Y. Djenouri, G. Srivastava, Y. Li, P. S. Yu, Scalable mining of high-utility sequential patterns with three-tier MapReduce model, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **16** (2021), 1–26. https://dx.doi.org/10.1145/3487046
60. D. Lo, S. C. Khoo, J. Li, Mining and ranking generators of sequential patterns, *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, (2008), 553–564. https://dx.doi.org/10.1137/1.9781611972788.51

61. J. M. Luna, F. Padillo, M. Pechenizkiy, S. Ventura, Apriori versions based on MapReduce for mining frequent patterns on big data, *IEEE T. Cybernetics*, **48** (2018), 2851–2865. https://dx.doi.org/10.1109/TCYB.2017.2751081

62. J. M. Luna, P. Fournier-Viger, S. Ventura, Frequent itemset mining: A 25 years review, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **9** (2019), e1329. https://doi.org/10.1002/widm.1329

63. C. Luo, S. M. Chung, Efficient mining of maximal sequential patterns using multiple samples, *Proceedings of the 2005 SIAM International Conference on Data Mining*, (2005), 415–426. SIAM. https://dx.doi.org/10.1137/1.9781611972757.37

64. N. R. Mabroukeh, C. I. Ezeife, A taxonomy of sequential pattern mining algorithms, *ACM Comput. Surv. (CSUR)*, **43** (2010), 1–41. https://dx.doi.org/10.1145/1824795.1824798

65. R. Manikandan, S. B. V. J. Sara, N. Yuvaraj, A. Chaturvedi, S. S. Priscila, M. Ramkumar, Sequential pattern mining on chemical bonding database in the bioinformatics field, *AIP Conference Proceedings*, **2393** (2022), 020050. https://dx.doi.org/10.1063/5.0074405

66. H. Mannila, H. Toivonen, A. Inkeri Verkamo, Discovery of frequent episodes in event sequences, *Data min. knowl. disc.*, **1** (1997), 259–289. https://dx.doi.org/10.1023/A:1009748302351

67. H. M. Marin-Castro, E. Tello-Leal, Event log preprocessing for process mining: A review, *Applied Sciences*, **11** (2021), 10556. https://dx.doi.org/10.3390/app112210556

68. F. Masseglia, P. Poncelet, R. Cicchetti, An efficient algorithm for web usage mining, *Networking and Information Systems Journal*, **2** (2000), 571–604.

69. I. Miliaraki, K. Berberich, R. Gemulla, S. Zoupanos, Mind the gap: Large-scale frequent sequence mining, *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, (2013), 797–808. https://dx.doi.org/10.1145/2463676.2465285

70. C. H. Mooney, J. F. Roddick, Sequential pattern mining – approaches and algorithms, *ACM Comput. Surv.*, **45** (2013), 1–39. https://dx.doi.org/10.1145/2431211.2431218

71. M. Muzammal, R. Raman, Mining sequential patterns from probabilistic databases, *Knowl. Inf. Syst.*, **44** (2015), 325–358. https://dx.doi.org/10.1007/s10115-014-0766-7

72. S. Nuruddin, M. Islam, M. Alam, J. Ovi, M. A. Islam, An efficient approach for sequential pattern mining on GPU using CUDA platform, *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, (2020), 1–9. https://dx.doi.org/10.1109/ISMSIT50672.2020.9255161

73. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research*, **12** (2011), 2825–2830.

74. J. Pei, J. Han, B. Mortazavi-Asl, H. Zhu, Mining access patterns efficiently from web logs, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (2000), 396–407. https://dx.doi.org/10.1007/3-540-45571-x_47

75. J. Pei, J. Han, W. Wang, Mining sequential patterns with constraints in large databases, *Proc. of the 11th Int'l Conf. on Information and Knowledge Management*, (2002), 18–25. https://dx.doi.org/10.1145/584792.584799

76. J. Pei, J. Han, W. Wang, Constraint-based sequential pattern mining: The pattern-growth methods, *J. Intel. Inf. Syst.*, **28** (2007), 133–160. https://dx.doi.org/10.1007/s10844-006-0006-z

77. T. T. Pham, Efficiently mining sequential generator patterns using prefix trees, *Fund. Inform.*, **138** (2015), 373–386. https://dx.doi.org/10.3233/FI-2015-1217

78. H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, U. Dayal, Multi-dimensional sequential pattern mining, *Proceedings of the tenth international conference on Information and knowledge management*, (2001), 81–88. https://dx.doi.org/10.1145/502585.502600

79. K. Poongodi, D. Kumar, Mining frequent serial positioning episode rules with forward and backward search technique from event sequences, *The Computer Journal*, (2022). https://dx.doi.org/10.1093/comjnl/bxac031 , bxac031

80. S. Qiao, C. Tang, S. Dai, M. Zhu, J. Peng, H. Li, Y. Ku, Partspan: Parallel sequence mining of trajectory patterns, *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, **5** (2008), 363–367. IEEE. https://dx.doi.org/10.1109/fskd.2008.33

81. C. Raissi, P. Poncelet, M. Teisseire, SPEED: mining maximal sequential patterns over data streams, *2006 3rd International IEEE Conference Intelligent Systems*, (2006), 546–552. IEEE. https://dx.doi.org/10.1109/IS.2006.348478

82. S. Rathore, S. Dawar, V. Goyal, D. Patel, Top-k high utility episode mining from a complex event sequence, *Proceedings of the 21st international conference on management of data, computer society of India*, (2016).

83. P. Ravikumar, P. Likhitha, B. Venus Vikranth Raj, R. Uday Kiran, Y. Watanobe, K. Zettsu, Efficient discovery of periodic-frequent patterns in columnar temporal databases, *Electronics*, **10** (2021), 1478. https://doi.org/10.3390/electronics10121478

84. Ritika, S. K. Gupta, Mining transactional databases for frequent and high-utility fuzzy sequential patterns with time intervals, *IEEE Access*, **10** (2022), 71107–71119. https://dx.doi.org/10.1109/ACCESS.2022.3188307

85. K. K. Roy, M. H. H. Moon, M. M. Rahman, C. F. Ahmed, C. K. Leung, Mining sequential patterns in uncertain databases using hierarchical index structure, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (2021), 29–41. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-75765-6_3

86. M. Sahli, E. Mansour, P. Kalnis, Parallel motif extraction from very long sequences, *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, (2013), 549–558. https://dx.doi.org/10.1145/2505515.2505575

87. A. Sallaberry, N. Pecheur, S. Bringay, M. Roche, M. Teisseire, Sequential patterns mining and gene sequence visualization to discover novelty from microarray data, *J. Biomed. Inform.*, **44** (2011), 760–774. https://dx.doi.org/10.1016/j.jbi.2011.04.002

88. A. Segatori, A. Bechini, P. Ducange, F. Marcelloni, A distributed fuzzy associative classifier for big data, *IEEE T. Cybernetics*, **48** (2018), 2656–2669. https://dx.doi.org/10.1109/TCYB.2017.2748225

89. S. Song, H. Hu, S. Jin, HVSM: a new sequential pattern mining algorithm using bitmap

representation, *International conference on advanced data mining and applications*, (2005), 455–463. Springer. https://dx.doi.org/10.1007/11527503_55

90. W. Song, W. Ye, P. Fournier-Viger, Mining sequential patterns with flexible constraints from mooc data, *Appl. Intell.*, **52** (2022), 16458–16474. https://doi.org/10.1007/s10489-021-03122-7
91. P. Songram, V. Boonjing, S. Intakosum, Closed multidimensional sequential pattern mining, *Third International Conference on Information Technology: New Generations (ITNG'06)*, (2006), 512–517. https://dx.doi.org/10.1109/ITNG.2006.41
92. H. K. Sowmya, N. V. Uma Reddy, C. Kavyashree, R. J. Anandhi, Discovery of frequent pagesets from weblog using Hadoop Mapreduce based parallel apriori algorithm, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, (2022), 765–770. https://dx.doi.org/10.23919/INDIACom54597.2022.9763104
93. R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '96)*, (1996), 1–17.
94. T. Truong, H. Duong, B. Le, P. Fournier-Viger, U. Yun, H. Fujita, Efficient algorithms for mining frequent high utility sequences with constraints, *Inform. Sciences*, **568** (2021), 239–264. https://dx.doi.org/10.1016/j.ins.2021.01.060
95. T. Truong-Chi, P. Fournier-Viger, A survey of high utility sequential pattern mining. In: Fournier-Viger P, Lin JCW, Nkambou R, Vo B, Tseng VS (eds) High-Utility Pattern Mining: Theory, Algorithms and Applications, (2019), 97–129. Springer International Publishing, Cham. https://dx.doi.org/10.1007/978-3-030-04921-8_4
96. C. F. Tsai, W. C. Lin, S. W. Ke, Big data mining with parallel computing: A comparison of distributed and MapReduce methodologies, *J. Syst. Software*, **122** (2016), 83–92. https://dx.doi.org/10.1016/j.jss.2016.09.007
97. C. Y. Tsai, B. H. Lai, A location-item-time sequential pattern mining algorithm for route recommendation, *Knowledge-Based Systems*, **73** (2015), 97–110. https://dx.doi.org/10.1016/j.knosys.2014.09.012
98. W. Van Der Aalst, Process mining, *Commun. ACM*, **55** (2012), 76–83. https://dx.doi.org/10.1145/2240236.2240257
99. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 30, 2017. Available from: `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`
100. J. Wang, J. Han, BIDE: efficient mining of frequent closed sequences, *Proc. of the 20th International Conference on Data Engineering*, (2004), 79–90. IEEE. https://dx.doi.org/10.1109/ICDE.2004.1319986
101. X. Wang, A. Hosseininasab, P. Colunga, S. Kadıoğlu, W. J. van Hoeve, Seq2Pat: Sequence-to-pattern generation for constraint-based sequential pattern mining, *Proc of the AAAI Conference on Artificial Intelligence*, **36** (2022), 12665–12671. https://dx.doi.org/10.1609/aaai.v36i11.21542

102. I. H. Witten, E. Frank, M. A. Hall, C. J. Pal, Data Mining: Practical Machine Learning Tools and Techniques, *Data Mining*, **2** (2005).

103. C. H. Wu, C. C. Lai, Y. C. Lo, An empirical study on mining sequential patterns in a grid computing environment, *Expert Syst. Appl.*, **39** (2012), 5748–5757. https://dx.doi.org/10.1016/j.eswa.2011.11.095

104. C. W. Wu, Y. F. Lin, P. S. Yu, V. S. Tseng, Mining high utility episodes in complex event sequences, *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2013), 536–544. https://dx.doi.org/10.1145/2487575.2487654

105. Y. Wu, Y. Tong, X. Zhu, X. Wu, NOSEP: Nonoverlapping sequence pattern mining with gap constraints, *IEEE T. Cybernetics*, **48** (2017), 2809–2822. https://dx.doi.org/10.1109/TCYB.2017.2750691

106. Y. Wu, C. Zhu, Y. Li, L. Guo, X. Wu, NetNCSP: Nonoverlapping closed sequential pattern mining, *Knowledge-based systems*, **196** (2020), 105812. https://dx.doi.org/10.1016/j.knosys.2020.105812

107. Y. Wu, L. Luo, Y. Li, L. Guo, P. Fournier-Viger, X. Zhu, X. Wu, NTP-Miner: Nonoverlapping three-way sequential pattern mining, *ACM T. Knowl. Discov. Data*, **16** (2021), 1–21. https://dx.doi.org/10.1145/3480245

108. Y. Wu, M. Chen, Y. Li, J. Liu, Z. Li, J. Li, X. Wu, ONP-Miner: One-off negative sequential pattern mining, *ACM T. Knowl. Discov. Data*, (2022). https://dx.doi.org/10.1145/3549940

109. Y. Wu, Q. Hu, Y. Li, L. Guo, X. Zhu, X. Wu, OPP-Miner: Order-preserving sequential pattern mining for time series, *IEEE T. on Cybernetics*, (2022). https://dx.doi.org/10.1109/TCYB.2022.3169327

110. X. Yan, J. Han, gspan: Graph-based substructure pattern mining, *2002 IEEE International Conference on Data Mining*, (2002), 721–724.

111. X. Yan, J. Han, R. Afshar, CloSpan: Mining closed sequential patterns in large datasets, *Proceedings of the 2003 SIAM international conference on data mining*, (2003), 166–177. https://dx.doi.org/10.1137/1.9781611972733.15

112. Z. Yang, M. Kitsuregawa, LAPIN-SPAM: An improved algorithm for mining sequential pattern, *21st International Conference on Data Engineering Workshops (ICDEW'05)*, (2005), 1222–1222. IEEE. https://dx.doi.org/10.1109/icde.2005.235

113. Z. Yang, Y. Wang, M. Kitsuregawa, LAPIN: Effective sequential pattern mining algorithms by last position induction for dense databases. In: Kotagiri R, Krishna PR, Mohania M, Nantajeewarawat E (eds) Advances in Databases: Concepts, Systems and Applications - DASFAA 2007, Springer Berlin Heidelberg, Berlin, Heidelberg, Lecture Notes in Computer Sciences, **4443** (2007), 1020–1023. https://dx.doi.org/10.1007/978-3-540-71703-4_95

114. J. Yin, Z. Zheng, L. Cao, Uspan: An efficient algorithm for mining high utility sequential patterns, *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, (2012), 660–668. https://dx.doi.org/10.1145/2339530.2339636

115. T. You, Y. Sun, Y. Zhang, J. Chen, P. Zhang, M. Yang, Accelerated frequent

closed sequential pattern mining for uncertain data, *Expert Syst. Appl.*, (2022), 117254. https://doi.org/10.1016/j.eswa.2022.117254

116. X. Yu, J. Liu, X. Liu, C. Ma, B. Li, A mapreduce reinforced distributed sequential pattern mining algorithm, *International Conference on Algorithms and Architectures for Parallel Processing*, (2015), 183–197. Springer. https://dx.doi.org/10.1007/978-3-319-27122-4_13
117. U. Yun, J. J. Leggett, WSpan: Weighted sequential pattern mining in large sequence databases, *2006 3rd international IEEE conference intelligent systems*, (2006), 512–517. IEEE. https://dx.doi.org/10.1109/IS.2006.348472
118. F. Zabihi, M. Ramezan, M. M. Pedram, A. Memariani, Fuzzy sequential pattern mining with sliding window constraint, *2010 2nd International Conference on Education Technology and Computer*, **5** (2010), V5–396–V5–400. https://dx.doi.org/10.1109/ICETC.2010.5530044
119. M. J. Zaki, Parallel sequence mining on shared-memory machines, *J. Parallel Distr. Com.*, **61** (2001), 401–426. https://dx.doi.org/10.1007/3-540-46502-2_8
120. M. J. Zaki, SPADE: An efficient algorithm for mining frequent sequences, *Mach. learn.*, **42** (2001), 31–60. https://dx.doi.org/10.1023/A:1007652502315
121. J. Zhang, Y. Wang, D. Yang, CCSpan: Mining closed contiguous sequential patterns, *Knowledge-Based Systems*, **89** (2015), 1–13. https://dx.doi.org/10.1016/j.knosys.2015.06.014
122. H. Zhu, P. Wang, X. He, Y. Li, W. Wang, B. Shi, Efficient episode mining with minimal and non-overlapping occurrences, *2010 IEEE International Conference on Data Mining*, (2010), 1211–1216. IEEE. https://dx.doi.org/10.1109/icdm.2010.25
123. X. Zhu, H. Deng, Z. Chen, A brief review on frequent pattern mining, *2011 3rd International Workshop on Intelligent Systems and Applications*, (2011), 1–4. https://dx.doi.org/10.1109/ISA.2011.5873451

AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)