Data Science and Pattern Recognition ©2017 ISSN XXXX-XXXX
Ubiquitous International Volume **1**, Number **1**, February **2017**

A Survey of Sequential Pattern Mining

Philippe Fournier-Viger
School of Natural Sciences and Humanities
Harbin Institute of Technology Shenzhen Graduate School
HIT Campus Shenzhen University Town Xili, Shenzhen, China
philfv@hitsz.edu.cn

Jerry Chun-Wei Lin
School of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School
HIT Campus Shenzhen University Town Xili, Shenzhen, China
jerrylin@ieee.org

Rage Uday Kiran
University of Tokyo, Tokyo, Japan
National Institute of Information and Communication Technology, Tokyo, Japan
uday_rage@tkl.iis.u-tokyo.ac.jp

Yun Sing Koh
Department of Computer Science
University of Auckland, Auckland, New Zealand
ykoh@cs.auckland.ac.nz

Rincy Thomas
Department of Computer Science and Engineering
SCT, Bhopal, India
rinc_thomas@rediffmail.com

Abstract. *Discovering unexpected and useful patterns in databases is a fundamental data mining task. In recent years, a trend in data mining has been to design algorithms for discovering patterns in sequential data. One of the most popular data mining tasks on sequences is sequential pattern mining. It consists of discovering interesting subsequences in a set of sequences, where the interestingness of a subsequence can be measured in terms of various criteria such as its occurrence frequency, length, and profit. Sequential pattern mining has many real-life applications since data is encoded as sequences in many fields such as bioinformatics, e-learning, market basket analysis, text analysis, and webpage click-stream analysis. This paper surveys recent studies on sequential pattern mining and its applications. The goal is to provide both an introduction to sequential pattern mining, and a survey of recent advances and research opportunities. The paper is divided into four main parts. First, the task of sequential pattern mining is defined and its applications are reviewed. Key concepts and terminology are introduced. Moreover, main approaches and strategies to solve sequential pattern mining problems are presented. Limitations of traditional sequential pattern mining approaches are also highlighted, and popular variations of the task of sequential pattern mining are presented. The paper also presents research opportunities and the relationship to other popular pattern mining problems. Lastly, the paper also discusses open-source implementations of sequential pattern mining algorithms.*
Keywords: Sequential pattern mining, Sequences, Frequent pattern mining, Itemset mining, Data Mining,

1. **Introduction.**

Data mining consists of extracting information from data stored in databases to understand the data and/or take decisions. Some of the most fundamental data mining tasks are clustering, classification, outlier analysis, and pattern mining [6, 58]. Pattern mining consists of discovering interesting, useful, and unexpected patterns in databases. This field of research has emerged in the 1990s with the seminal paper of Agrawal and Srikant [1]. That paper introduced the Apriori algorithm, designed for finding frequent itemsets, that is groups of items (symbols) frequently appearing together in a database of customer transactions. For example, the Apriori algorithm can be used to discover patterns such as $\{carrot\_juice, salad, kiwi\}$ in a retail store database, indicating that these products are frequently bought together by customers.

The interest in pattern mining techniques comes from their ability to discover patterns that can be hidden in large databases and that are interpretable by humans, and hence useful for understanding the data and for decision-making. For example, a pattern $\{milk, chocolate\_cookies\}$ can be used to understand customer behavior and take strategic decisions to increase sales such as co-promoting products and offering discounts.

Although pattern mining has become very popular due to its applications in many domains, several pattern mining techniques such as those for frequent itemset mining [1, 53, 116, 86, 106] and association rule mining [1] are aimed at analyzing data, where the sequential ordering of events is not taken into account. Thus, if such pattern mining techniques are applied on data with time or sequential ordering information, this information will be ignored. This may result in the failure to discover important patterns in the data, or finding patterns that may not be useful because they ignore the sequential relationship between events or elements. In many domains, the ordering of events or elements is important. For example, to analyze texts, it is often relevant to consider the order of words in sentences [94]. In network intrusion detection, the order of events is also important [93].

To address this issue, the task of *sequential pattern mining* was proposed. It is a prominent solution for analyzing sequential data [2, 98, 117, 4, 51, 89, 3, 47, 30, 111, 31, 32, 27, 28, 22, 100, 79]. It consists of discovering interesting subsequences in a set of sequences, where the interestingness of a subsequence can be measured in terms of various criteria such as its occurrence frequency, length, and profit. Sequential pattern mining has numerous real-life applications due to the fact that data is naturally encoded as sequences of symbols in many fields such as bioinformatics [108], e-learning [22], market basket analysis [98], text analysis [94], energy reduction in smarthomes [104], webpage click-stream analysis [25] and e-learning [124]. Moreover, sequential pattern mining can also be applied to time series (e.g. stock data), when discretization is performed as a pre-processing step [66]

Sequential pattern mining is a very active research topic, where hundreds of papers present new algorithms and applications each year, including numerous extensions of sequential pattern mining for specific needs. Because of this, it can be difficult for newcomers to this field to get an overview of the field. To address this issue, a survey has been published in 2010 [79]. However, this survey is no longer up-to-date as it does not discuss the most recent techniques, advances and challenges in the field. In this paper, we aim to address this issue by presenting an up-to-date survey of sequential pattern mining that can serve both as an introduction and as a guide to recent advances and opportunities in the field. The rest of this paper is organized as follows. The next section describes the problem of sequential pattern mining, and the main techniques employed in sequential pattern mining. Then, the paper discusses popular extensions of the problem of sequential pattern mining, and other problems in data mining that are closely related to sequential pattern mining. Then, the paper discusses research opportunities and open-source implementations of sequential pattern mining algorithms. Finally, a conclusion is drawn.

2. **Sequential Pattern Mining.**

Two types of sequential data are commonly used in data mining [58]: time-series and sequences. A *time-series* is an ordered list of numbers, while a *sequence* is an ordered list of nominal values (symbols). For example, Fig. 1 (left) shows a time-series representing amounts of money, while Fig. 1 (right) depicts a sequence of symbols (letters). Both time-series and sequences are used in many domains. For instance, time-series are often used to represent data such as stock prices, temperature readings, and electricity consumption readings, while sequences are used to represent data such as sentences in texts (sequences of words), sequences of items purchased by customers in retail stores, and sequences of webpages visited by users.

The problem of sequential pattern mining was proposed by Agrawal and Srikant [98], as the problem of mining interesting subsequences in a set of sequences. Although, it was originally designed to be applied to sequences, it can also be applied to time series after converting time-series to sequences using discretization techniques. For example, Fig. 1 (right) shows a sequence representing the time-series shown in Fig. 1 (left), where the symbols a, b, c, d are defined as an increase of 10\$, a decrease of 10\$, an increase of 20\$, and a decrease of 20\$, respectively. There exists several ways of transforming time-series to sequences. Some of the most popular techniques are the SAX [66] and iSAX [17] algorithms. For more details about time-series transformations, the reader may refer to a survey of methods for time-series data mining [23].

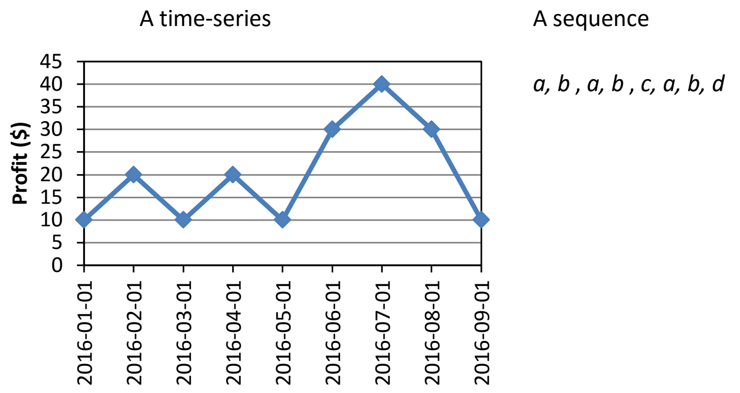


Figure 1. A time-series (left) and a sequence (right)

In this paper, we are interested by sequences, as it is the type of data used in sequential pattern mining. Definitions related to sequences are given next with some illustrative examples. Let there be a set of items (symbols) $I = \{i_1, i_2, \dots i_m\}$. An *itemset* X is a set of items such that $X \subseteq I$. Let the notation $|X|$ denote the set cardinality or, in other words, the number of items in an itemset X. An itemset X is said to be of length k or a k-itemset if it contains k items ($|X| = k$). For example, consider that the set of symbols $I = \{a, b, c, d, e, f, g\}$ represents the products sold in a retail store. The set $\{a, b, c\}$ is an itemset containing three items, which may represent the purchases made by a customer on a given day.

Without loss of generality, assume that there exists a total order on items $\prec$ such as the lexicographical order (e.g. $a \prec b \prec c \prec d \prec e \prec f \prec g$). A *sequence* is an ordered list of itemsets $s = \langle I_1, I_2, ..., I_n \rangle$ such that $I_k \subseteq I$ ($1 \leq k \leq n$). For example, consider the sequence $\langle \{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\} \rangle$ representing five transactions made by a customer at a retail store. Each single letter represents an item. Items between curly brackets represent an itemset. This sequence indicates that a customer purchased items a and b at the same time, then bought item c, then purchased items f and g at the same time, then purchased g, and finally purchased e.

A sequence $s_a = \langle A_1, A_2, \dots, A_n \rangle$ is said to be *of length* k or a k*-sequence* if it contains k items, or in other words if $k = |A_1| + |A_2| + \cdots + |A_n|$. For example, the sequence $\langle \{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\} \rangle$ is a 7-sequence.

A *sequence database SDB* is a list of sequences $SDB = \langle s_1, s_2, ..., s_p \rangle$ having sequence identifiers (SIDs) $1, 2...p$. For instance, a sequence database is shown in Table 1, which contains four sequences having the SIDs 1, 2, 3 and 4. These sequences could, for example, represent purchases made by four customers.

Table 1. A sequence database

SID	**Sequence**
1	$\langle \{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\} \rangle$
2	$\langle \{a, d\}, \{c\}, \{b\}, \{a, b, e, f\} \rangle$
3	$\langle \{a\}, \{b\}, \{f, g\}, \{e\}$
4	$\langle \{b\}, \{f, g\} \rangle$

A sequence $s_a = \langle A_1, A_2, ..., A_n \rangle$ is said to be *contained in* another sequence $s_b = \langle B_1, B_2, ..., B_m \rangle$ if and only if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{in}$ (denoted as $s_a \sqsubseteq s_b$). For example, the sequence $\langle \{b\}, \{f, g\} \rangle$ is contained in sequence $\langle \{a, b\}, \{c\}, \{f, g\}, \{g\}, \{e\} \rangle$, while the sequence $\langle \{b\}, \{g\}, \{f\} \rangle$ is not. If a sequence s_a is contained in a sequence s_b, s_a is said to be a *subsequence* of s_b.

The goal of sequential pattern mining is to discover interesting subsequences in a sequence database, that is sequential relationships between items that are interesting for the user. Various measures can be

TABLE 2. Sequential patterns found in the database of Table. 1

Pattern	**Sup.**	**Closed?**	**Maximal?**	**Generator?**
$\langle\{a\}\rangle$	3	yes	no	no
$\langle\{a\},\{g\}\rangle$	2	no	no	yes
$\langle\{a\},\{g\},\{e\}\rangle$	2	yes	yes	no
$\langle\{a\},\{f\}\rangle$	3	yes	no	no
$\langle\{a\},\{f\},\{e\}\rangle$	2	yes	yes	no
$\langle\{a\},\{c\}\rangle$	2	no	no	yes
$\langle\{a\},\{c\},\{f\}\rangle$	2	yes	yes	no
$\langle\{a\},\{c\},\{e\}\rangle$	2	yes	yes	no
$\langle\{a\},\{b\}\rangle$	2	no	no	yes
$\langle\{a\},\{b\},\{f\}\rangle$	2	yes	yes	no
$\langle\{a\},\{b\},\{e\}\rangle$	2	yes	yes	no
$\langle\{a\},\{e\}\rangle$	3	yes	no	yes
$\langle\{a,b\}\rangle$	2	yes	yes	no
$\langle\{b\}\rangle$	4	no	no	yes
$\langle\{b\},\{g\}\rangle$	3	yes	no	no
$\langle\{b\},\{g\},\{e\}\rangle$	2	yes	yes	no
$\langle\{b\},\{f\}\rangle$	4	yes	no	no
$\langle\{b\},\{f,g\}\rangle$	2	yes	yes	no
$\langle\{b\},\{f\},\{e\}\rangle$	2	yes	yes	no
$\langle\{b\},\{e\}\rangle$	3	yes	no	no
$\langle\{c\}\rangle$	2	no	no	yes
$\langle\{c\},\{f\}\rangle$	2	no	no	yes
$\langle\{c\},\{e\}\rangle$	2	no	no	yes
$\langle\{e\}\rangle$	3	no	no	yes
$\langle\{f\}\rangle$	4	no	no	yes
$\langle\{f,g\}\rangle$	2	no	no	yes
$\langle\{f\},\{e\}\rangle$	2	no	no	yes
$\langle\{g\}\rangle$	3	no	no	yes
$\langle\{g\},\{e\}\rangle$	2	no	no	yes

used to assess how interesting a subsequence is. In the original problem of sequential pattern mining, the *support* measure is used. The *support* (or *absolute support*) of a sequence s_a in a sequence database SDB is defined as the number of sequences that contain s_a, and is denoted by $sup(s_a)$. In other words, $sup(s_a) = |\{s|s \sqsubseteq s_a \wedge s \in SDB\}|$. For example, the support of the sequence $\langle\{b\},\{f,g\}\rangle$ in the database of Table 1 is 2 because this sequence appears in two sequences (Sequence 1 and Sequence 4). Note that some papers define the support of a sequence X as a ratio. This definition called the *relative support* is $relSup(s_a) = sup(s_a)/|SDB|$, that is the number of sequences containing s_a divided by the number of sequences in the database. For example, the relative support of the itemset $\langle\{b\},\{f,g\}\rangle$ is 0.5.

Sequential pattern mining is the task of finding all *frequent subsequences* in a sequence database. A sequence s is said to be a *frequent sequence* or a *sequential pattern* if and only if $sup(s) \geq minsup$, for a threshold *minsup* set by the user [98]. The assumption is that frequent subsequences are interesting to the user. For instance, consider the database of Table 1, and assume that the user sets $minsup = 2$ to find all subsequences appearing in at least two sequences. Table 2 shows the 29 sequential patterns found in the database for $minsup = 2$, and their support values. For example, the patterns $\langle\{a\}\rangle$ and $\langle\{a\},\{g\}\rangle$ are frequent and have a support of 3 and 2 sequences, respectively.

The task of sequential pattern mining is an enumeration problem. It aims at enumerating all patterns (subsequences) that have a support no less than the minimum support threshold set by the user. Thus, there is always a single correct answer to a sequential pattern mining problem. Discovering sequential patterns is a hard problem. To solve this problem, the naive approach is to calculate the support of all possible subsequences in a sequence database to then output only those meeting the minimum support constraint specified by the user. However, such a naive approach is inefficient because the number of subsequences can be very large. A sequence containing q items in a sequence database can have up to $2^q - 1$ distinct subsequences. Because of this, applying the naive approach to solve the sequential pattern mining problem is unrealistic for most real-life sequence databases. Hence, it is necessary to design efficient algorithms to avoid exploring the search space of all possible subsequences.

Numerous algorithms have been designed to discover sequential patterns in sequence databases. Some of the most popular are GSP [98], Spade [117], PrefixSpan [89], Spam [3], Lapin [111], CM-Spam [30], and CM-Spade [30]. All these sequential pattern mining algorithms take as input a sequence database and a minimum support threshold (chosen by the user), and output the set of frequent sequential patterns. It is important to note that there is always only one correct answer to a sequential pattern mining task (for a given sequence database and threshold value). Thus, all sequential pattern mining algorithms return the same set of sequential patterns if they are run with the same parameter on the same database. Hence, the difference between the various algorithms is not their output, but it is how each algorithm discovers the sequential patterns. Various algorithms utilize different strategies and data structures to search for sequential patterns efficiently. As a result, some algorithms are more efficient than others.

In the following pages, we give an overview of the main techniques employed by sequential pattern mining algorithms, and discusses their advantages and limitations. The following section then discusses variations of the sequential pattern mining problem.

Before introducing the specific techniques, it is important to present a few key definitions related to the search for sequential patterns. In general, sequential pattern mining algorithms assume that there exists a total order on items denoted as $\succ$, which represent the order for processing items in a database to find the sequential patterns. For example, consider a database containing the items $\{a, b, c\}$. The order for processing items could be defined as the lexicographical order (e.g. $c \succ b \succ a$). But any other total order on items from I such as the order of decreasing or increasing support could be used. Note that the choice of the order $\succ$ has no influence on the final result produced by a sequential pattern mining algorithm. The order $\succ$ is used so that algorithms follow a specific order to explore potential sequential patterns, and thus avoid considering the same pattern more than once.

All sequential pattern mining algorithms explore the search space of sequential patterns by performing two basic operations called *s-extensions* and *i-extensions*. These operations are used to generate a $(k+1)$-sequence (a sequence containing $k+1$ items) from a k-sequence. These operations are defined as follows. A sequence $s_a = \langle A_1, A_2, ..., A_n \rangle$ is a *prefix* of a sequence $s_b = \langle B_1, B_2, ..., B_m \rangle$, if $n < m$, $A_1 = B_1, A_2 = B_2, ..., A_{n-1} = B_{n-1}$ and A_n is equal to the first $|A_n|$ items of B_n according to the $\succ$ order. For example, the sequence $\langle \{a\} \rangle$ is a prefix of the sequence $\langle \{a, b\}, \{c\} \rangle$, and the sequence $\langle \{a\}\{c\} \rangle$ is a prefix of the sequence $\langle \{a\}, \{c, d\} \rangle$ A sequence s_b is said to be an *s-extension* of a sequence $s_a = \langle I_1, I_2, ...I_h \rangle$ with an item x, if $s_b = \langle I_1, I_2, ...I_h, \{x\} \rangle$, i.e. s_a is a prefix of s_b and the item x appears in an itemset occuring after all the itemsets of s_a. For example, the sequence $\langle \{a\}, \{a\} \rangle$ and $\langle \{a\}, \{b\} \rangle$ and $\langle \{a\}, \{c\} \rangle$ are s-extensions of the sequence $\langle \{a\} \rangle$. A sequence s_c is said to be an *i-extension* of s_a with an item x, if $s_c = \langle I_1, I_2, ...I_h \cup \{x\} \rangle$, i.e. s_a is a prefix of s_c and the item x is appended to the last itemset of s_a, and the item x is the last one in I_h according to the $\succ$ order. For example, the sequences $\langle \{a, b\} \rangle$ and $\langle \{a, c\} \rangle$ are i-extensions of the sequence $\langle \{a\} \rangle$.

In general, sequential pattern mining algorithms can be categorized as being either depth-first search or breadth-first search algorithms. *Breadth-first search* algorithms such as GSP proceed as follows. They first scan the database to find frequent 1-sequences (sequential patterns containing a single item). Then, they generate 2-sequences by performing s-extensions and i-extensions of 1-sequences, then 3-sequences are generated using the 2-sequences, then 4-sequences are generated using the 3-sequences, and so on until no sequences can be generated. This approach is also called a *level-wise approach* since patterns are considered in ascending order of their length. For example, consider a sequence database containing three items $I = \{a, b, c\}$. A breadth-first algorithm will first consider the 1-sequences $\langle \{a\} \rangle$, $\langle \{b\} \rangle$, and $\langle \{c\} \rangle$. Then, the algorithm will consider the 2-sequences $\langle \{a\}, \{a\} \rangle$, $\langle \{a\}, \{b\} \rangle$, $\langle \{a\}, \{c\} \rangle$, $\langle \{a, b\} \rangle$, $\langle \{a, c\} \rangle$, $\langle \{b\}, \{a\} \rangle$, $\langle \{b\}, \{b\} \rangle$, $\langle \{b\}, \{c\} \rangle$, $\langle \{b, c\} \rangle$, $\langle \{c\}, \{a\} \rangle$, $\langle \{c\}, \{b\} \rangle$ and $\langle \{c\}, \{c\} \rangle$
Then, the algorithm will consider the 3-sequences, 4-sequences, and so on until no patterns can be generated. It can be observed that the search space can be very large, as there are many different ways to combine items to generate a potential sequential pattern. Assume that the longest sequence in a database contains x items. A breadth-first search sequential pattern mining algorithm will explore in the worst case all possible sequences containing x items or less. If a database contains m items, this number can be greater than 2^m.

Depth-first search algorithms such as Spade [117], PrefixSpan [89], Spam [3], Lapin [111], CM-Spam [30], and CM-Spade [30] explore the search space of patterns by following a different order. They start from the sequences containing single items (e.g. $\langle \{a\} \rangle$, $\langle \{b\} \rangle$, and $\langle \{c\} \rangle$), and then recursively perform i-extensions and s-extensions with one of these sequences to generate larger sequences. Then, when a pattern can no longer be extended, the algorithm backtrack to generate other patterns using other sequences. For example, consider a sequence database containing the items $I = \{a, b, c\}$, and consider that only sequential patterns containing no more than three items are considered (for the purpose of having a small example). A depth-first search algorithm assuming the lexicographical order (e.g. $c \succ b \succ a$), which performs i-extensions before s-extensions will explore potential sequential patterns following this order: $\langle\rangle$, $\langle\{a\}\rangle$, $\langle\{a,b\}\rangle$, $\langle\{a,b,c\}\rangle$, $\langle\{a,c\}\rangle$, $\langle\{a\},\{a\}\rangle$, $\langle\{a\},\{a,b\}\rangle$, $\langle\{a\},\{a,c\}\rangle$, $\langle\{a\},\{b\}\rangle$, $\langle\{a\},\{b,c\}\rangle$, $\langle\{a\},\{b\},\{c\}\rangle$, $\langle\{a\},\{c\}\rangle$, $\langle\{a\},\{c\},\{a\}\rangle$, $\langle\{a\},\{c\},\{b\}\rangle$, $\langle\{a\},\{c\},\{c\}\rangle$, $\langle\{b\}\rangle$, $\langle\{b,c\}\rangle$, $\langle\{b\},\{a\}\rangle$, $\langle\{b\},\{a,b\}\rangle$, $\langle\{b\},\{a,c\}\rangle$, $\langle\{b\},\{a\},\{a\}\rangle$, $\langle\{b\},\{a\},\{b\}\rangle$, $\langle\{b\},\{a\},\{c\}\rangle$, $\langle\{b\},\{b\}\rangle$, $\langle\{b\},\{b,c\}\rangle$, $\langle\{b\},\{b\},\{a\}\rangle$, $\langle\{b\},\{b\},\{b\}\rangle$, $\langle\{b\},\{b\},\{c\}\rangle$, $\langle\{b\},\{c\}\rangle$, $\langle\{b\},\{c\},\{a\}\rangle$, $\langle\{b\},\{c\},\{b\}\rangle$, $\langle\{b\},\{c\},\{c\}\rangle$, $\langle\{c\}\rangle$, $\langle\{c\},\{a\}\rangle$, $\langle\{c\},\{a,b\}\rangle$, $\langle\{c\},\{a,c\}\rangle$, $\langle\{c\},\{a\},\{a\}\rangle$, $\langle\{c\},\{a\},\{b\}\rangle$, $\langle\{c\},\{a\},\{c\}\rangle$, $\langle\{c\},\{b\}\rangle$, $\langle\{c\},\{b,c\}\rangle$, $\langle\{c\},\{b\},\{a\}\rangle$, $\langle\{c\},\{b\},\{b\}\rangle$, $\langle\{c\},\{b\},\{c\}\rangle$, $\langle\{c\},\{c\}\rangle$, $\langle\{c\},\{c\},\{a\}\rangle$, $\langle\{c\},\{c\},\{b\}\rangle$, and $\langle\{c\},\{c\},\{c\}\rangle$.

Since the search space of all possible sub-sequences in a database can be very large, designing an efficient algorithm for sequential pattern mining requires to integrate techniques to avoid exploring the whole search space. The basic mechanism for pruning the search space in sequential pattern mining is based on the following property called the *Apriori property*, *downward-closure property* or *anti-monotonicity property*. This property states that for any sequence s_a and s_b, if s_a is a subsequence of s_b ($s_a \sqsubset s_b$), then s_b must have a support that is lower or equal to the support of s_a. For example, consider two sequences $\langle\{a\}\rangle$ and $\langle\{a,b\}\rangle$. It is obvious that the support (the occurrence frequency) of $\langle\{a,b\}\rangle$ cannot be greater than the support of $\langle\{a\}\rangle$ since $\langle\{a,b\}\rangle$ is more specific than $\langle\{a\}\rangle$. It is thus said that the support measure is *monotonic*. The above property is useful for pruning the search space since if a sequence is infrequent, if follows that all its extensions are also infrequent, and thus are not sequential patterns. For example, consider the database of Table 1 and assume that $minsup = 2$. Since the pattern $\langle\{c\},\{g\}\rangle$ is infrequent, all its extensions such as $\langle\{c\},\{g\},\{e\}\rangle$ are also infrequent and hence can be ignored. The application of the downward-closure property can thus greatly reduce the search space of sequential patterns.

In general, sequential pattern mining algorithms differ in (1) whether they use a depth-first or breadth-first search, (2) the type of database representation that they use internally or externally, (3) how they generate or determine the next patterns to be explored in the search space, and (4) how they count the support of patterns to determine if they satisfy the minimum support constraint.

AprioriAll is the first sequential pattern mining algorithm [2]. The authors of AprioriAll then proposed an improved version called GSP [98]. The AprioriAll and GSP algorithms are inspired by the Apriori algorithm for frequent itemset mining [1].

GSP uses a standard database representation, as shown in Table 1, also called a *horizontal database*. The GSP algorithm performs a level-wise search to discover frequent sequential patterns. It first scans the database to calculate the support of all 1-sequences. Then, it keeps all frequent 1-sequences in memory. For example, consider the sequence database of Table 1. The frequent 1-sequences are $\langle\{a\}\rangle$, $\langle\{b\}\rangle$, $\langle\{c\}\rangle$, $\langle\{e\}\rangle$, $\langle\{f\}\rangle$, and $\langle\{g\}\rangle$. Then, the GSP algorithm recursively explores larger patterns. During this search, GSP uses the sequential patterns of length k to generates potentially frequent patterns of length $k+1$. This process of generating candidates is done by combining pairs of patterns of length k that share all but one item. For example, in the above example, GSP will combine the 1-sequential patterns to obtain 2-sequences: $\langle\{a,b\}\rangle$, $\langle\{a,c\}\rangle$, $\langle\{a,e\}\rangle$, $\langle\{a,f\}\rangle$, $\langle\{a,g\}\rangle$, $\langle\{a\},\{a\}\rangle$, $\langle\{a\},\{b\}\rangle$, $\langle\{a\},\{c\}\rangle$, $\langle\{a\},\{e\}\rangle$, $\langle\{a\},\{f\}\rangle$, $\langle\{a\},\{g\}\rangle$, $\langle\{b,c\}\rangle$, $\langle\{b,d\}\rangle$, $\ldots\langle\{g\},\{g\}\rangle$. After generating candidates, the GSP algorithm will evaluate each candidate to determine if it is a sequential pattern (if it has a support no less than the *minsup* threshold), to identify patterns that should be output. This is done in two steps. First, for a candidate pattern s_a, GSP will check if all its sub-sequences of length k-1 are also frequent. If s_a has a subsequence that is not frequent, s_a cannot be frequent (it would violate the downward-closure property). Otherwise, GSP will scan the database to calculate the support of s_a. If s_a is frequent, it will be output. In this example, GSP finds that the frequent 2-sequences are: $\langle\{a,b\}\rangle$, $\langle\{a\},\{b\}\rangle$, $\langle\{a\},\{c\}\rangle$, $\langle\{a\},\{e\}\rangle$, $\langle\{a\},\{f\}\rangle$, $\langle\{a\},\{g\}\rangle$, $\langle\{b\},\{e\}\rangle$, $\langle\{b\},\{f\}\rangle$, $\langle\{b\},\{g\}\rangle$, $\langle\{c\},\{e\}\rangle$, $\langle\{c\},\{f\}\rangle$, $\langle\{f,g\}\rangle$, $\langle\{f\},\{e\}\rangle$, and $\langle\{g\},\{e\}\rangle$. Then the GSP algorithm continues this process to generate sequential patterns of length 3, 4, and so on, until no pattern can be generated. The final set of sequential patterns is shown in Table 2. The GSP algorithm is well-known since it is one of the first sequential pattern mining algorithms. In recent years, many algorithms have been shown to be more efficient than GSP. The reason is that GSP has several important limitations:

- **Multiple database scans.** One of the main problems of GSP is that it repeatedly scans the database to calculate the support of candidate patterns. This can be very costly for large database, even if some optimizations may be performed to reduce that cost (e.g. by sorting sequences by their size to avoid comparing long patterns with short sequences).

- **Non-existent candidates.** Another problem of GSP is that it may generate patterns that do not exist in the database. The reason is that GSP generates candidates by combining smaller patterns without accessing the database. Hence, GSP can waste time considering many patterns that are non-existent in the database. For example, the pattern $\langle\{g\},\{g\}\rangle$ would be considered as a potential sequential pattern by GSP for the database of Table 1, although it does not appear in this database.
- **Maintaining candidates in memory.** Another serious problem of the GSP algorithm is typical of breadth-first search pattern mining algorithms. It is that at any moment it must keep all frequent sequences of length k in memory to be able to generate patterns of length $k+1$. This can consume a huge amount of memory.

The Spade [117] algorithm is an alternative algorithm that utilizes a depth-first search, and avoid some of the drawbacks of the GSP algorithm. The Spade algorithm is inspired by the Eclat [116] algorithm for frequent itemset mining. It utilizes a *vertical database representation* rather than an horizontal database representation. The vertical representation of a sequence database indicates the itemsets where each item i appears in the sequence database [117, 3, 30]. For a given item, this information is called the *IDList* of the item. For example, Fig. 2 shows the vertical representation of the horizontal database of Figure 1. In this example, the IDList of item g indicates that g occurs in the third and fourth itemsets of sequence 1, and in the second itemset of sequence 4. The vertical representation of an horizontal database (the IDLists of all single items) can be created by scanning the horizontal database once. Note that it is also possible to perform the reverse process to create an horizontal database from a vertical database (the difference between an horizontal and vertical reprensentation of a database is just how the information is stored).

Formally, the *IDList* of an item or pattern is defined as follows. Let there be a pattern $s_a = \langle A_1, A_2, ..., A_n\rangle$ appearing in a sequence $s_b = \langle B_1, B_2, ..., B_m\rangle$ $(s_a \sqsubseteq s_b)$. The item-positions of s_a in s_b, denoted as $ip(s_a, s_b)$, is the set of pairs of the form (s_a, ik), where ik is an integer such that $1 \leq i1 < i2 < ... < ik \leq m$ and $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_n \subseteq B_{ik}$. For example, the item-positions of the pattern $\langle\{a\}\rangle$ in the sequence s_2 of Table 1 is $\{(s_2, 1), (s_2, 4)\}$, indicating that the pattern $\langle\{a\}\rangle$ appears in the first and fourth itemsts of sequence s_2. The *IDList* of a pattern s_a for a sequence database SDB is defined as the list of all item-positions of s_a in all sequences where it appears, that is $IDList(s_a) = \bigcup_{s_a \sqsubseteq s_b \wedge s_b \in SDB} ip(s_a, s_b)$. For example, the IDList of $\langle\{a\}\rangle$ (also depicted in Fig. 2) is $IDList(s_a) = \{(s_1, 1), (s_2, 1), (s_2, 4), (s_3, 1)\}$.

A vertical database has two interesting properties for sequential pattern mining. The first property is that the IDList of any pattern allows to directly calculate its support. The support of a pattern s_a is simply the number of distinct sequence identifiers in its IDList. For example, the IDList of the pattern $\langle\{a, b\}\rangle$ is $\{(s_1, 1), (s_2, 3), (s_2, 4), (s_3, 2)\}$. The number of distinct sequence identifiers in this IDList is 3. Thus, the support of this patterns is 3. The second property is that the IDList of any pattern s_a obtained by performing an i-extension or s-extension of a pattern s_b with an item i can be created without scanning the original database by joining the IDList of s_b with the IDList of the item i. For example, by comparing the IDLists of patterns $\langle\{a\}\rangle$ and $\langle\{b\}\rangle$, it is possible to obtain the IDList of the pattern $\langle\{a, b\}\rangle$. The detailed process of this join operation is not presented here. The interested reader may refer to the article describing the Spade [117] algorithm for more details.

a	
SID	Itemsets
1	1
2	1,4
3	1
4	

b	
SID	Itemsets
1	1
2	3,4
3	2
4	1

c	
SID	Itemsets
1	2
2	2
3	
4	

d	
SID	Itemsets
1	
2	1
3	
4	

e	
SID	Itemsets
1	5
2	4
3	4
4	

f	
SID	Itemsets
1	3
2	4
3	3
4	2

g	
SID	Itemsets
1	3,4
2	
3	3
4	2

FIGURE 2. A vertical database

By utilizing the above properties, algorithms such as Spade [117], Spam[3], CM-Spam [30], and CM-Spade [30] explore the whole search space of patterns by reading the database only once to create the

IDLists of single items. Then, the IDLists of any pattern encountered when browsing the search space is obtained by performing the join of IDLists, which allows to calculate the support of the pattern. Thus, all frequent patterns can be enumerated without repeatedly scanning the database, and without maintaining a large number of patterns in memory (contrarily to breadth-first search algorithms). As a result, this approach was shown to be one of the most efficient for sequential pattern mining, and to greatly outperform the first algorithms, who adopted a breadth-first search approach.

The IDList representation is used in several sequential pattern mining algorithms. A popular optimization of the IDList structure used in the Spam [3] and BitSpade [4] algorithms is to encode IDLists as bit vectors. The motivation for this optimization is that IDLists can be very large when patterns appear in many sequences (especially, in dense databases or databases containing long sequences), and that applying the join operation of IDLists is costly as it requires to compare the elements of two IDLists. The solution introduced in the Spam [3] algorithm is to represent IDLists as bit vectors. The bit vector representation of an IDList is defined as follows. Let SDB be a sequence database containing k items and m sequences, where $size(i)$ denotes the number of itemsets in the i-th sequence of SDB. Consider a pattern s_a and its IDList $IDList(s_a)$. The bit vector representation of this IDList, denoted as $BList(s_a)$ is a bitvector containing $\sum_{i=1}^{m} size(i)$ bits, where the j-th bit represents the p-th itemset of the t-th sequence of SDB, such that $\sum_{i=1}^{min(0,t-1)} size(i) < j < \sum_{i=1}^{t} size(i)$ and $p = j - \sum_{i=1}^{min(0,t-1)}$. The j-th bit is set to 1 if $(s_t, p) \in IDList(s_a)$, and otherwise it is set to 0. For example, the bit vector representation of the IDLists of single items is shown in Table 3.

TABLE 3. The bitvector representation of the vertical database

Item x	**IDList of x as a bit vector**
a	100001001100000
b	100000011010010
c	010000100000000
d	000001000000000
e	000010001000100
f	001000001001001
g	001100000001001

Using bitvectors to represent IDLists can greatly reduce the amount of memory used for mining sequential patterns with a vertical representation. This is especially true for dense datasets where many bits are generally set to 1 in IDLists. For sparse datasets, many bits are set to 0, and thus the amount of saved memory is less. Note that it is possible to use various bit vector compression techniques to further compress the bit vectors. For example, some bit vector compression techniques have been used in various pattern mining algorithm implementations to ignore some bits set to zero [35]. The version of the Spade algorithm using bit vectors is called BitSpade [4]. Another algorithm that is similar to BitSpade and also performs a depth-first search using bit vector IDLists is Spam [3]. Moreover, an algorithm inspired by Spam called Fast [100] introduced the concept of indexed sparse IDLists, to more quickly calculate the support of candidates and reduce memory usage. A version of Spade using bit vectors called Prism also introduced the concept of Prime-block encoding [47]. It was shown that the Spam, BitSpade, and Prism, which uses the bit vector representation, are more than an order of magnitude faster than the original Spade algorithm [4, 3], while the Fast algorithm was shown to be faster than Spam [100] but was not compared with Spade or the improved BitSpade algorithm.

Recently, the Spam [3] and BitSpade [117] algorithms were improved to obtain the CM-Spam and CM-Spade algorithms [30]. These algorithms are based on the observations that Spam and Spade generate many candidate patterns and that performing the join operation to create the IDList of each of them is costly. To reduce the number of join operations, the CM-Spam and CM-Spade algorithm introduced the concept of *co-occurrence pruning* [30]. It consists of initially scanning the database to create a structure called the *Co-occurrence Map* (CMAP) that stores all frequent 2-sequences. Then, for each pattern s_a that is considered by the search procedure if the two last items of s_a are not a frequent 2-sequences, the pattern s_a can be directly eliminated without constructing its IDList (thus without performing the join operation). It was shown that CM-Spam and CM-Spade outperformed GSP, Spam, BitSpade [117], and PrefixSpan by more than an order of magnitude. CM-Spade is claimed to be the current fastest algorithm [30].

Besides breadth-first search algorithms and vertical algorithms, another important type of algorithms for sequential pattern mining are the *pattern-growth algorithms*. These algorithms are depth-first search algorithms, designed to address a limitation of the previously described algorithms, which is to generate candidate patterns that may not appear in the database. The reason why the previously described algorithms may generate patterns not appearing in the database is that they generate candidate patterns by combining smaller patterns but this process does not involve accessing the database.

Pattern-growth algorithms avoid this problem by recursively scanning the database to find larger patterns. Thus, they only consider the patterns actually appearing in the database. Performing database scans can however be costly. To reduce the cost of database scans, pattern-growth algorithms have introduced the concept of *projected database* [53, 86, 89], which aims at reducing the size of databases as larger patterns are considered by the depth-first search.

The most popular pattern-growth algorithm for sequential pattern mining is PrefixSpan [89], which draws inspiration from the FPGrowth algorithm for itemset mining [53]. PrefixSpan proceeds as follows. It explores the search space of sequential patterns using a depth-first search. It starts from sequential patterns containing a single item and explores larger patterns by recursively appending items to patterns to create larger patterns. To ensure that no patterns are generated twice, the items are appended to patterns according to a total order on items $\prec$, which can be the lexicographical order or any other total order. The first operation performed by PrefixSpan is to scan the original sequence database to calculate the support of single items an identify the frequent items (those having a support that is no less than the *minsup* threshold). Then, PrefixSpan outputs each of these items as a frequent sequential patterns, and consider these patterns as seeds to pursue the depth-first search. During the depth-first search, for a given sequential pattern s_a of length k, PrefixSpan first creates the projected database of the pattern s_a. Then, PrefixSpan scans the projected database of s_a to count the support of items to find items that can be appended to s_a by i-extension or s-extension to form $(k+1)$-sequential patterns. This process is then recursively repeated as a depth-first search to find all frequent sequential patterns.

Table 4. The projected sequence database of pattern $\langle\{a\}\rangle$

SID	**Sequence**
1	$\langle\{\_, b\}, \{c\}, \{f, g\}, \{g\}, \{e\}\rangle$
2	$\langle\{\_, d\}, \{c\}, \{b\}, \{a, b, e, f\}\rangle$
3	$\langle\{b\}, \{f, g\}, \{e\}$
4	$\langle\{b\}, \{f, g\}\rangle$

Table 5. The projected sequence database of pattern $\langle\{a, b\}\rangle$

SID	**Sequence**
1	$\langle\{c\}, \{f, g\}, \{g\}, \{e\}\rangle$
2	$\langle\{\_, e, f\}\rangle$

We illustrate these steps with a short example. Consider that $minsup = 2$. By scanning the database of Fig. 1, PrefixSpan will find that the frequent 1 sequences are $\langle\{a\}\rangle$, $\langle\{b\}\rangle$, $\langle\{c\}\rangle$, $\langle\{e\}\rangle$, $\langle\{f\}\rangle$, and $\langle\{g\}\rangle$. These sequences will thus be output. Then, assuming the lexicographical ordering of items, PrefixSpan will first consider the item a to try to find larger frequent sequences starting with the prefix $\langle\{a\}\rangle$. PrefixSpan will thus scan the original database to build the projected database of $\langle\{a\}\rangle$, shown in Table 4. The projected database of the pattern $\langle\{a\}\rangle$ is the set of sequences where the pattern $\langle\{a\}\rangle$ appears, but where all items and itemsets appearing before the first occurrence of $\langle\{a\}\rangle$ have been removed [89]. Then, to find frequent sequential patterns that starts with the prefix $\langle\{a\}\rangle$ containing one more item, the Prefixspan algorithm will read the projected database of $\langle\{a\}\rangle$ and count the support of all items appearing in that database that could be appended either by i-extension or s-extension to $\langle\{a\}\rangle$. For instance, the 2-sequences that are i-extensions or s-extensions of $\langle\{a\}\rangle$ are: $\langle\{a, b\}\rangle : 2$, $\langle\{a, d\}\rangle : 1$, $\langle\{a\}, \{a\}\rangle : 1$, $\langle\{a\}, \{b\}\rangle : 3$, $\langle\{a\}, \{c\}\rangle : 2$, $\langle\{a\}, \{e\}\rangle : 3$, $\langle\{a\}, \{f\}\rangle : 4$, and $\langle\{a\}, \{g\}\rangle : 2$, where for each pattern, the number after the colon (:) indicates its support. Then, PrefixSpan will output the sequential patterns (those having a support greater than or equal to 2), that is: $\langle\{a, b\}\rangle$, $\langle\{a\}, \{b\}\rangle$, $\langle\{a\}, \{c\}\rangle$, $\langle\{a\}, \{e\}\rangle$, $\langle\{a\}, \{f\}\rangle$, and $\langle\{a\}, \{g\}\rangle$, Then, PrefixSpan will continue its depth-first exploration of the search space by attempting to find sequential patterns starting with the prefix $\langle\{a, b\}\rangle$. PrefixSpan will scan the database of the pattern $\langle\{a\}\rangle$ to create the projected database of $\langle\{a, b\}\rangle$. This database is depicted in Table 5. While creating the projected database of $\langle\{a, b\}\rangle$, PrefixSpan will count the support of items that can extend $\langle\{a, b\}\rangle$ by i-extension of s-extension. This process will then continue in the same way (by pursuing the depth-first search) until all sequential patterns have been found.

The pattern-growth approach of PrefixSpan has the advantage that it only explores patterns appearing in the database (unlike many other sequential pattern mining algorithms). However, a drawback of PrefixSpan and other pattern-growth algorithms is that it can be costly to repeatedly scan the database and create database projections, in terms of runtime. Moreover, in terms of memory, creating database projections can consume a huge amount of memory if it is naively implemented, as in the worst case it requires to copy almost the whole database for each database projection. In the PrefixSpan algorithm an optimization called *pseudo-projection* is used to reduce this cost, which consists of implementing a projected database as a set of pointers on the original database [89, 86]. Another notable pattern-growth sequential pattern mining algorithms is FreeSpan [51], an early version of PrefixSpan, proposed by the same research team.

The time complexity of sequential pattern mining algorithms depends on the number of patterns in the search space, and the cost of the operations for generating and processing each itemset. Pattern-growth algorithms have the advantage over other algorithms of only considering patterns that actually appear in the database. Thus, it would seem reasonable to expect that they would be faster than other algorithms. However, in practice, it has been reported that this is not the case. The CM-Spade algorithm was for example reported to outperform PrefixSpan by more than an order of magnitude [30]. The reason is that the cost of scanning the database and performing projections can be quite high.

The number of patterns in the search space depends on how the *minsup* threshold is set by the user and on how similar the sequences are in a sequence database. In general, as the *minsup* threshold is decreased the number of sequential patterns found by sequential pattern mining algorithms can increase exponentially. The search space for sequential pattern mining can be very large even for small sequence databases containing a few sequences. For example, a sequence database containing only two identical sequences of 100 items can contain 2^{100} sequential patterns.

In the above paragraphs, we have discussed the three main types of sequential pattern mining algorithms: breadth-first algorithms that perform candidate generation (e.g. AprioriAll, and GSP), depth-first search algorithms that perform candidate generation using the IDList structure and its variations (e.g. Spade, Spam, BitSpade, Fast, CM-Spam, CM-Spade), and pattern-growth algorithms (e.g. FreeSpan, PrefixSpan). Most sequential pattern mining algorithms extends these three main approaches.

3. **Variations of the Sequential Pattern Mining Problem.** The task of sequential pattern mining has many applications. Nonetheless, for some applications, it also has some fundamental limitations. To address this issue, several extensions or variations of the problem of sequential pattern mining have been proposed. This section aims at providing and up-to-date review of some of the most popular variations/extensions.

A first important limitation of the traditional problem of sequential pattern mining is that a huge number of patterns may be found by the algorithms, depending on a database's characteristics and how the *minsup* threshold is set by users. Finding too many patterns is an issue because users typically do not have much time to analyze a large amount of patterns. Moreover, as more patterns are found, the algorithms' performance typically decrease in terms of memory and runtime. To address this issue, a solution that has been extensively studied is to discover *concise representations of sequential patterns* instead of all sequential patterns [30, 31, 32, 28, 108, 109, 45, 44, 67, 64, 63, 46, 70, 92, 114, 48, 56]. A concise representation is a subset of all sequential patterns that is meaningful and summarize the whole set of sequential patterns.

Several concise representations of sequential patterns have been proposed and several algorithms have been designed to directly discover these representations without extracting all sequential patterns. It was shown that these algorithms can be order of magnitudes faster than traditional sequential pattern mining algorithms and find a much smaller set of patterns. Some of these concise representations were also shown to provide higher classification accuracy compared to using all sequential patterns for classification tasks [46, 92]. Let FS denotes the set of all sequential patterns. There are three main concise representations of sequential patterns.

- *Closed sequential patterns* [56, 48, 30, 108, 109] are the set of sequential patterns that are not included in other sequential patterns having the same support, that is: $CS = \{s_a | s_a \in FS \wedge \nexists s_b \in FS$ such that $s_a \sqsubset s_b \wedge sup(s_a) = sup(s_b)\}$. For example, consider the sequence database of Table 1. Among the 29 sequential patterns found in this database for $minsup = 2$, only 16 of them are closed sequential patterns (identified in Table 2). Discovering closed sequential patterns instead of all sequential patterns thus considerably reduce the result set presented to the user. The closed sequential patterns are interesting because they are a *lossless* representation of all sequential patterns, that is using the closed sequential patterns, it is possible to recover all the sequential patterns and their support without accessing the database. Another reason why the closed sequential patterns are interesting is that they represents the largest subsequences common to sets of sequences. For example, in market basket analysis, if each sequence represents a customer, the closed patterns represent the largest patterns common to groups of customers. The first algorithms for closed sequential pattern mining are CloSpan [109] and Bide [108], which are pattern-growth algorithms extending PrefixSpan. Recent algorithms such as Clasp [48], CloFast [39] and CM-Clasp [30] adopt a vertical representation and were shown to outperform early algorithms.

- *Maximal sequential patterns* [31, 28, 45, 44, 67, 64, 63] are the set of sequential patterns that are not included in other sequential patterns, that is: $MS = \{s_a | s_a \in FS \wedge \nexists s_b \in FS$ such that $s_a \sqsubset s_b\}$. For example, Table 2 shows that among all the 29 sequential patterns found in the database of Table 1, only 10 are maximal. An interesting property of maximal patterns is that $(MS \subseteq CS \subseteq FS)$. In other words, the set of maximal patterns is always not larger than the set of closed sequential patterns and all sequential patterns. In practice, the number of maximal patterns can be of several orders of magnitude less than closed patterns or all patterns. However, maximal sequential patterns are not lossless. The maximal patterns may be used to obtain all sequential patterns without scanning the database, but their support can only be recovered by performing an additional database scan [31]. Several algorithms have been proposed for maximal sequential pattern mining [31, 28, 45, 44, 67, 64, 63], including breadth-first search algorithms (e.g. AprioriAdjust [63]), pattern-growth algorithms (e.g. MaxSP [28]), and vertical algorithms (e.g. VMSP [31]). Moreover, approximate algorithms have also been proposed such as DIMASP [45]. Maximal sequential patterns have several applications such as to find the frequent longest common subsequences to sentences in texts, analysing DNA sequences, data compression and web log mining [45].
- *Generator sequential patterns (aka sequential generators)* [32, 46, 70, 92, 114] are the set of sequential patterns that have no subsequence having the same support, that is: $GS = \{s_a | s_a \in FS \wedge \nexists s_b \in FS$ such that $s_b \sqsubset s_a \wedge sup(s_a) = sup(s_b)\}$. For example, 14 sequential patterns are generators in the database of Table 1, for $minsup = 2$ (depicted in Table 2[1]). The set of sequential generators is a subset of all sequential patterns, but can be larger, equal or smaller than the set of closed patterns [70]. However, it can be argued that generators are preferable to other representations according to the MDL (Minimum Description Length) principle [9] as generators are the smallest subsequences that characterize group of sequences in a sequence database [32, 70, 92]. Additionally, generators can be combined with closed patterns to generate rules with a minimum antecedent and a maximum consequent, which allows deriving the maximum amount of information based on the minimum amount of information [32]. This can be useful for example in market basket analysis [70]. Other usage of generators is for classification, where they were shown to provide higher accuracy than using all patterns or closed patterns [46, 92]. Pattern-growth algorithms for mining sequential generators are GenMiner [70], FEAT [46] and FSGP [114]. A more recent algorithm named VGEN [32], which extends the CM-Spam algorithm, was shown to ouperform FEAT and FSGP.

Algorithms for discovering concise representations of patterns are generally inspired by similar work on itemset mining such as for the discovery of closed itemsets [65, 106, 120, 83, 7, 107], maximal itemsets [106], and generator itemsets [101, 33, 103]. Several techniques and theoretical results from these works have been adapted to sequential pattern mining. However, there are also some key differences between concise representation of itemsets and sequential patterns. Some of the key differences are that in sequential pattern mining, the set of generator patterns is not guaranteed to be a subset of the set of closed patterns, and multiple closed patterns may correspond to a same set of sequences [70]. Algorithms for mining concise representations generally adopt a *candidate-maintenance-and-test* approach, which consists of exploring the search space of sequential patterns and keeping candidate patterns that may belong to the desired concise representations in memory, until the final result is obtained (e.g. CloSpan, VGEN, VMSP). Other algorithms, which are said to be *without candidate generation* (e.g. BIDE and MaxSP), perform database scans to directly output patterns rather than keeping potential candidates in memory.

[1]Note that the empty sequence $\langle\rangle$ is also considered to be a sequential generator, with a support of 4 sequences, although it is not shown in Table 2.

To reduce the number of sequential patterns found and find more interesting patterns, researchers have also proposed to integrate constraints in sequential pattern mining [91]. A constraint is an additional set of criteria that the user provides to indicate more precisely the types of patterns to be found. Numerous kinds of constraints have been studied. There are two ways of applying constraints. The first way is to apply them as a post-processing step on the set of all sequential patterns to filter uninteresting patterns. However, a problem with this approach is that enumerating all sequential patterns can consume a lot of time and requires a huge amount of memory. The second way to address this problem is to push the constraints deep in the mining process. In other words, the constraints are applied during the search for patterns to reduce the search space. Algorithms adopting this approach can be orders of magnitude faster, and generate much less patterns than traditional sequential pattern mining algorithms, depending on the constraints used.

One of the first sequential pattern mining algorithm to integrate constraints is GSP [98]. It introduced the constraints of minimum and maximum amount of time between two consecutive itemsets in sequential patterns (*gap constraints*), as well as a maximum time duration for each sequential pattern (*duration constraint*). The consideration of time constraints has also been the subject of the Hirate and Yamana algorithm [113] and Fournier-Viger algorith [22], which have extended the PrefixSpan and BIDE algorithm with gap and duration constraints. The integration of gap constraints in vertical algorithms was done in Pex-Spam [54] by extending the Spam algorithm. Pei et al. studied the integration of various constraints in pattern-growth algorithms such as items that should appear or not in sequential patterns (*item constraints*), minimum/maximum number of items per sequential patterns (*length constraints*), and aggregate constraints on prices of items in sequences such as average, minimum, maximum, sum and standard deviation of prices (*aggregate constraints*) [91]. Another type of constraints considered in sequential pattern mining is regular expressions (*regular expression constraints*). The SPIRIT [42] algorithm lets users specify regular expressions on patterns to be found. It converts constraints to an automaton for pruning patterns when performing a breadth-first search.

Pei et al. [91] and other researchers studied the characteristics of constraints that can be pushed deep into the process of mining sequential patterns, and other types of patterns [87, 88, 10]. Three main types of constraints have been identified. *Anti-monotone constraints* such as the minimum support threshold and length constraints, gap constraints and duration constraints are some of the easiest and most beneficial to integrate in a pattern mining algorithm, as they can be used to prune the search space by applying the downward closure property. *Convertible constraints* are constraints that are neither monotone nor anti-monotone but that can be converted to anti-monotone constraints if some additional strategies are applied [87]. A *succinct constraint* is a constraint that can be checked for a pattern by only looking at the single items that it contains. For example, the constraint that the sum of the weights of a sequential pattern should be not greater or not less than a given value can be checked by simply adding the weights of its items. This constraint is both succint and anti-monotone. For more information about the use of constraints, the reader may refer to the referenced papers [91, 87, 88, 10].

Another limitation of traditional sequential pattern mining studies is that they solely focus on the discovery of items that are positively correlated in sequence databases. However, for some applications, negative correlations are more interesting than positive correlations. To address this limitation, the problem of mining *negative sequential patterns* has been studied [118, 57, 119, 20]. A negative sequential pattern is a pattern containing the negation of a least one item. Mining negative sequential patterns is more difficult than mining only positive patterns as the search space becomes larger. To mine positive and/or negative patterns, the Negative-GSP [118] and PNSP [57] algorithms were proposed, extending GSP. Then, a genetic algorithm was proposed, which was shown to outperform these two algorithms [119], as well as a SPADE-based algorithm named e-NSP [20].

Another interesting extension of the problem of sequential pattern mining is *multi-dimensional sequential pattern mining* [22, 85, 99]. It considers an extended type of sequence database where each sequence can be annotated with dimensions having symbolic values. For example, in the context of market basket analysis, a database of customer purchase sequences could be annotated with three dimensions: gender, education level, and income. Then, a multi-dimensional sequential pattern mining algorithm can discover sequential patterns that are common to various dimension values. For example, a pattern could be discovered with the dimension values $(male, university, *)$ indicating that it is common to some male customers having a university degree but with any income. To mine multi-dimensional patterns, there are two main approaches: mining the dimensions using an itemset mining algorithm and then the sequential patterns, or mining the sequential patterns and then mining the dimensions [22, 85, 99].

Another limitation of traditional sequential pattern mining algorithms is that they assume that sequence databases are static. In fact, traditional sequential pattern mining algorithms are said to be *batch algorithms* as they are designed to be applied once to a sequence database to obtain patterns. Then, if the database is updated, algorithms need to be run again from scratch to obtain the updated patterns. This approach is inefficient because sometimes only small changes are made to a database that would not require to perform the whole search for patterns again. As a solution to this problem, several *incremental sequential pattern mining algorithms* have been designed [13, 78, 82, 76, 74, 75]. To our best knowledge, the first algorithm is ISM [84], which adopts an IDList approach inspired by SPADE to update sequential patterns when new sequences are inserted in a sequence database (*sequence insertion*). Another algorithm is IncSpan [13], which is based on the PrefixSpan algorithm. However, this algorithm was found to rely on an incorrect property, and was fixed by Nguyen et al. [82]. Another recent algorithm named PreFUSP-TREE-INs [74] is based on the pre-large concept, which consists of keeping a buffer of almost frequent sequential patterns in memory to avoid scanning the database when few sequences are inserted. Recently, algorithms have been also designed to handle *sequence modifications* [76] and *sequence deletions* [75]. Among incremental algorithms, a few of them have also been designed to interactively mine patterns by for example considering that the user may change the parameters of the algorithm and perform multiple queries involving constraints [84]. Another interactive algorithm is KISP, which extends the GSP algorithm [62].

A particular type of incremental algorithms called *stream mining algorithms* have been designed to mine sequential patterns in a potentially infinite stream of sequences [55, 96, 14, 69]. They assume that sequences may arrive at a very high speed and may be only read once. They are approximate algorithms that are designed to process each sequence as quickly as possible. To our best knowledge, the first algorithm for mining sequential patterns in streams is eISeq [14]. However, this algorithm considers a simple type of sequence database where sequences of items are taken as input rather than sequences of itemsets. The IncSPAM [55] algorithm is designed for the general case. It extends the SPAM algorithm [55]. Algorithms were also designed for mining concise representations of patterns in streams. For example, SPEED [96] and Seqstream [15] mine maximal and closed sequential patterns in data streams, respectively. Generally, stream mining algorithms consider a recency constraint to discover sequential patterns that are recently recent (but may be infrequent in the past).

Another extension of sequential pattern mining is top-k sequential pattern mining [29]. It consists of discovering the k most frequent sequential patterns in a sequence database. The rationale for this problem is that it is often difficult for user to set the *minsup* threshold using traditional sequential pattern mining algorithms if the user has no background knowledge about the database. If the *minsup* threshold is set too low, too many patterns may be found and the algorithms may become very slow, and if the *minsup* threshold is set too high, too few patterns may be found. Top-k sequential pattern mining algorithms address this problem by letting the users directly indicate the number of patterns k to be found instead of using the *minsup* parameter. Top-k sequential pattern mining is a harder problem than sequential pattern mining [29].

Some other extensions of sequential pattern mining extends the sequence database representation in various ways to extract more rich patterns. The following paragraphs reviews some of the most popular extensions.

- *Weighted sequential pattern mining* is an extension of sequential pattern mining where weights (generally assumed to be in the [0,1] interval) are associated to each item to indicate their relative importance [18, 112, 97]. The goal of weighted sequential pattern mining is to find sequential patterns that have a minimum weight.
- *High-utility sequential pattern mining* (HUSPM) is an extension of weighted sequential pattern mining where not only item weights are considered but also item quantities in sequences [5, 115, 72, 8]. The traditional problem of sequential pattern mining takes as input a sequence database where purchase quantities are binary, that is each item appears in an itemset of a sequence or not. For several applications, this assumption does not hold. For example, consider a sequence database of customer transactions where customers may have bought zero, one, or several units of each product. Not considering the purchase quantities may lead to discovering misleading patterns. To address this limitation, HUSP generalizes the problem of sequential pattern mining by considering that each item appears zero, once or multiple times in each itemset (purchase quantities), and that each item has a weight indicating its relative importance (e.g. how much profit is generated by each unit sold of the item). The goal of HUSP is to find all sequential patterns that have a *utility* greater than or equal to a minimum utility threshold in a sequence database. The utility (profit) of a sequential pattern is the sum of the maximum utility (profit) generated by the pattern in each sequences where it appears [5, 115, 72, 8]. HUSP is quite challenging as the utility measure is neither monotone nor anti-monotone unlike the support measure traditionally used in sequential

pattern mining. Thus, the utility measure cannot be directly used to prune the search space. To address this issue, HUSP algorithm have introduced upper-bounds on the utility of sequential patterns such as the SWU [5] measure that are monotone, to prune the search space. A major challenge in HUSP has been to develop tighter upper-bounds on the utility to be able to prune a larger part of the search space, and improve the performance of HUSP [5, 115, 72, 8]. HUSP is a very active research topic. Various extensions of the HUSP problem have been studied such as to hide high utility sequential patterns in databases to protect sensitive information [95] and discovering high-utility sequential rules [125].

- *Uncertain sequential pattern mining* is an extension of sequential pattern mining that considers data uncertainty [80, 81, 121, 49, 50]. For many applications, data stored in databases is uncertain for various reasons such that data has been collected using noisy sensors or that the data is inaccurate or imperfect. Two main models have been designed to discover sequential patterns that frequently appears in uncertain sequence databases [80]. The first model is called the *expected-support model* [80]. According to this model, each item i appearing in an itemset X is annotated with an existence probability $pr(i, X)$ representing the certainty that this item appeared in the itemset (a value in the [0,1] interval). The expected-support of a sequential pattern s_a in a sequence is defined as the product of the expected-support of its items in the sequence if it appears in the sequence, and otherwise 0. The expected-support of a sequential pattern s_a in a database SDB is the sum of its expected-support values for all sequences where s_a appears[2]. The task of uncertain sequential pattern mining in the expected support model is to discover all sequential patterns that are expected to be frequent. A variation of this model also considers source uncertainty (whether a sequence can be attributed to a given source instead of other(s)) [80]. The second model is the *probabilistic sequential pattern mining model* [81]. To use this model, the user has to set two thresholds: a minimum confidence threshold *minprob* and a minimum support threshold *minsup*. A sequential pattern is then considered a *probabilistic sequential pattern* if the calculated probability that it appears in more than *minsup* transactions by considering possible worlds is greater than *minprob*. Various algorithm have been proposed for uncertain sequential pattern mining. Algorithms based on GSP, SPAM and PrefixSpan were presented for the expected support model [81]. A PrefixSpan based algorithm named SeqU-PrefixSpan was proposed for the probabilistic sequential pattern mining model [121]. In a variation of the problem of uncertain sequential pattern mining [49], the uncertainty of timestamps for events was considered. Besides, designing scalable algorithms for uncertain sequential pattern mining using the Spark big data framework has also been studied [50].
- *Fuzzy sequential pattern mining* is another important extension of sequential pattern mining [21, 52]. It considers databases where items in sequences takes quantitative values (in the [0,1] interval), and where fuzzy membership functions are used to map these values to nominal values. For example, some items *chocolate bar* and *soft drink* in a sequence could be annotated with values of 0.5 and 0.7 representing their respective sweetness, which could be translated to nominal values such as *sweet* and *very sweet*. To compute the support in a fuzzy sequence database, various algorithms have been proposed [21, 52] based on different ways of counting the occurrence of patterns in fuzzy sequences. For instance, the SpeedyFuzzy and MiniFuzzy algorithms count a pattern as appearing in a sequence if (1) the membership values of all its items are greater than 0, or (2) if they are greater than some threshold, respectively [21]. Numerous studies about discovering other types of fuzzy patterns such as fuzzy itemsets have also been published [16, 73].

4. **Other pattern mining problems.** Besides sequential pattern mining, several other pattern mining problems have been studied. Research on these problems have inspired research on sequential pattern mining. This sections reviews some of the most important related problems.

- *Itemset mining* is the task of discovering frequent itemsets [1, 53, 116, 86, 106] in a transaction database. It can be seen as a special case of the sequential pattern mining problem where each sequence contains a single itemset. Thus, all items are assumed to be simultaneous, and there is no sequential ordering between items. Formally, a transaction database D is a set of transactions, where each transaction T is an unordered set of items. An itemset X is a set of items. The support of an itemset X is the number of transactions that contain the itemset, that is $sup(X) = |\{T|X \subseteq T \wedge T \in D\}|$. Given a minimum support threshold *minsup* set by the user, the goal of frequent itemset mining is to find all *frequent itemsets*, that is all itemsets having a support no less

[2]Note that if a pattern is allowed to appear multiple times in a sequence, this definition needs to be generalized.

than *minsup*. Numerous extensions of the problem of itemset mining are similar to variations of the sequential pattern mining problem such as high-utility itemset mining [71, 36, 126], uncertain itemset mining [121, 49, 50], fuzzy itemset mining [16, 73], and stream itemset mining [12, 102].

- *Association rule mining* (ARM) [1, 26, 68] consists of finding association rules in a transaction database. ARM also does not consider the sequential ordering of items. An *association rule* is a pattern of the form $X \rightarrow Y$ where X and Y are two itemsets such that $X \cap Y = \emptyset$. Association rules are evaluated using interestingness measures. The two traditional measures used in association rule mining are the *support* ($sup(X \rightarrow Y) = sup(X \cup Y)$) and *confidence* ($conf(X \rightarrow Y) = sup(X \cup Y)/sup(X)$). The support of a rule measures how often it appears in a database, while the confidence can be seen as a measure of the conditional probability $P(Y|X)$. To find association rules, a user has to provide two thresholds: a minimum support threshold and a minimum confidence threshold. The result of ARM is the set of all rules having a support and confidence respectively no less than these thresholds. An advantage of association rules over frequent itemsets is that association rules not only assess how frequently items co-occur but also if there is a strong association between them. It is interesting to note that besides the support and confidence, more than 20 other interestingness measures have been proposed in the literature. The interested reader may refer to Lenca et al. [68] for a survey of association rule measures. Generating association rules is generally done in two phases: mining frequent itemsets in a transaction database, and then using these itemsets to generate the rules [1].
- *Sequential rule mining* is a variation of the sequential pattern mining problem where *sequential rules* [38, 34] of the form $X \rightarrow Y$ are discovered, indicating that if some items X appear in a sequence it will be followed by some other items Y with a given confidence. The concept of a sequential rule is similar to that of association rules excepts that it is required that X must appear before Y according to the sequential ordering, and that sequential rules are mined in sequences rather than transactions. Sequential rules address an important limitation of sequential pattern mining, which is that although some sequential patterns may appear frequently in a sequence database, the patterns may have a very low confidence and thus be worthless for decision-making or prediction. For example, consider the database of Table 1. The sequential pattern $\langle (f)(e) \rangle$ is considered frequent if $minsup = 2$ because this pattern appears in 2 sequences. Thus, it may be tempting to think that f is likely to be followed by e in other sequences. However, this is not the case. By looking at Table 1, it can be found that f is actually followed by e in only two of the four sequences where f appears. This example shows that sequential patterns can be misleading. Sequential rules addresses this problem by not only considering their support but also their confidence. For example, the sequential rule $\{f\} \rightarrow \{e\}$ has a support of 2 sequences and a confidence of 50%, indicating that although this rule is frequent, it is not a strong rule. Formally, the confidence of a sequential rule $X \rightarrow Y$ is defined as the number of sequences containing the items X before the items Y divided by the number of sequences containing the items X [38]. Numerous sequential rule mining have been proposed such as RuleGrowth [38] and ERMiner [34], which respectively adopt a pattern-growth and a vertical approach for discovering rules. Moreover, several variations of the problem have been proposed for example to mine sequential rules with a window constraint [38], mine high-utility sequential rules [125] and to discover the top-k most frequent sequential rules [24]. Sequential rules have been reported as more effective than sequential patterns for some tasks involving prediction [38]. Sequential rule mining has numerous applications such as e-learning, web page prefetching, anti-pattern detection, alarm sequence analysis and restaurant recommendation [34].
- *Episode mining* is another pattern mining problem that shares many similarities with sequential pattern mining. The main difference is that episode mining aims at finding patterns in a single sequence rather than a set of sequences. The goal of episode mining is to find itemsets (sets of items) frequently appearing in a sequence (*frequent episodes*) or to discover rules of the form $X \rightarrow Y$ (*episode rules*), indicating that X often appears before Y within a time window set by the user. Epsiode mining can be used to analyze various types of data such as web-click streams, telecommunication data, sensor readings, sequences of events on an assembly line and network traffic data [77, 122].
- *Periodic pattern mining* [41, 105, 60, 61] consists of finding patterns that appear frequently and periodically in a single sequence. The periodicity of a pattern is measured based on its period lengths. The period lengths of a pattern are the time elapsed between any two consecutive occurrences of the pattern. To discover periodic patterns, a user has to specify constraints on period lengths such as a minimum and maximum average period length, and a maximum period length [105]. Applications

of periodic pattern mining are numerous, and include stock market analysis, market analysis and bioinformatics [41].

- *Sub-graph mining* [110, 59, 11] is another popular pattern mining problem. The goal of sub-graph mining is to find all *frequent sub-graphs* in a large graph or a database of graphs. To mine sub-graphs, the user must specify a minimum support threshold. Then, a sub-graph mining algorithms outputs all graphs appearing frequently. Unlike sequential pattern mining, the traditional problem of sub-graph mining do not consider the time dimension. Graph mining is a quite challenging task because the search space can be very large even for small graph databases and it is necessary to design strategies to check if different generated graphs are isomorphic [110]. Various extensions of the problem of sub-graph mining have been studied such as closed and maximal sub-graph mining [59]. Sub-graph mining has various applications such as the analysis of chemical compounds [110, 59, 11].

5. **Large Research opportunities.** The problem of sequential pattern mining and other related problems have been studied for more than two decades, and are still very active research areas. There are many possibilities for further research on sequential pattern mining. Below, we list some important research opportunities.

- *Applications.* An important research opportunity is to utilize sequential pattern mining in new applications, or in new ways for existing applications. Since sequential data occurs in many fields, there are many ways that sequential pattern mining can be applied. One of the most interesting and promising possibility is to apply sequential pattern mining in emerging research fields such as the internet of things, social network analysis and sensor networks.
- *Developing more efficient algorithms.* Sequential pattern mining is computationally expensive in terms of runtime and memory. This can a problem for dense databases, or databases containing numerous sequences or long sequences, depending on the *minsup* threshold chosen by the user. For this reason, many studies have been done on improving the performance of sequential pattern mining algorithms by developing novel data structures and algorithms, and introducing constraints. Although current algorithms are much faster than the first algorithms, there is still a need for improvement. Besides, more research should be carried on designing efficient algorithms for variations of the sequential pattern mining problem such as uncertain sequential pattern mining and high-utility sequential pattern mining. Some promising areas of research are the design of parallel, distributed, multi-core, and GPU-based algorithms. In general, depth-first search algorithms are easier to parallelize than breadth-first search algorithms.
- *Designing algorithms to handle more complex data.* Another interesting research opportunity is to extend sequential pattern mining algorithms so that they can be applied to sequence databases containing more complex types of data. Some extensions have been mentioned in this paper. However, there are still many possibilities. This research is important since it will allows sequential pattern mining to solve new real-world problems. Some recent researches have for example considered the spatial dimension [19].
- *Designing algorithms for finding more complex and meaningful patterns.* Another imported issue is to find more complex patterns in sequences. Furthermore, to find more meaningful patterns, research should be further carried to develop interestingness measures and evaluation methods for evaluating the usefulness of the patterns found [68]. This would lead to finding more interesting and useful patterns.

6. **Open-source Implementations.** Although sequential pattern mining has been studied for more than two decades, an important issue is that the majority of researchers in this field do not release their implementations or source code. This is problematic because other researchers often need to re-implement algorithms from other researchers to be able to use their algorithms or compare their performance with novel algorithms. But when a researcher implement the algorithm of another researcher, there always remains a doubt that the implementation may not be as efficient as the original algorithm. Besides, even when binary files are released it has been noted in studies such as the one of Goethal [43] that the performance of pattern mining algorithms may vary greatly depending on the compiler used and the machine architecture used for running performance comparison.

The solution to the above issue is that more researchers share implementations of their algorithms. To our best knowledge, the largest collection of open-source implementations of sequential pattern mining algorithms is the *SPMF data mining library* [35, 40] (`http://www.philippe-fournier-viger.com/spmf/`). It provides more than 120 algorithms for discovering patterns in databases such as sequential patterns, sequential rules, periodic patterns, itemsets and association rules. The SPMF library is developed in Java and is multi-platform. Its source code is released under the GPL3 license. It is designed to be easily integrated in other Java software programs, and can be run as a standalone software using its command-line or graphical user interface. Datasets used for benchmarking sequential pattern mining algorithms can be found on the SPMF website.

Another important issue related to the public release of algorithm implementations is that many researchers do not compare the performance of new algorithms with the previous best algorithms. It is recommended that researchers proposing new algorithms compare their performance with the previous best algorithms.

7. **Conclusion.** This paper has provided a detailled survey of sequential pattern mining. The paper has presented the main types of algorithms for discovering sequential patterns. Moreover, the paper has presented important extensions of the sequential pattern mining problems that address some shortcomings of sequential pattern mining. In addition, the paper has discussed other research problem related to sequential pattern mining such as itemset mining, association rule mining, sequential rule mining and periodic pattern mining. Finally, the paper has discussed research opportunities and open-source implementations of sequential pattern mining algorithms.

Acknowledgment. This work is supported by the Youth 1000 Talent funding from the National Science Fundation of China, and an initiating fund from the Harbin Institute of Technology.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *The International Conference on Very Large Databases*, pp. 487–499, 1994.

[2] R. Agrawal, and R. Srikant, "Mining sequential patterns," *The International Conference on Data Engineering*, pp. 3–14, 1995.

[3] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 429–435, 2002.

[4] S. Aseervatham, A. Osmani, and E. Viennet, "bitSPADE: A lattice-based sequential pattern mining algorithm using bitmap representation," *The International Conference on Data Mining*, pp. 792–797, 2006.

[5] C. F. Ahmed, S. K. Tanbeer, and B. S. Jeong, "A novel approach for mining high-utility sequential patterns in sequence databases," *Electronics and Telecommunications Research Institute journal*, vol. 32(5), pp. 676–686, 2010.

[6] C. C. Aggarwal, *Data mining: the textbook*, Heidelberg:Springer, 2015.

[7] G. Aliberti, A. Colantonio, R. Di Pietro, and R. Mariani, "EXPEDITE: EXPress closED ITemset enumeration," *Expert Systems with Applications*, vol. 42(8), pp. 3933–3944, 2015.

[8] O. K. Alkan, and P. Karagoz, "CRoM and HuspExt: improving efficiency of high utility sequential pattern extraction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27(10), pp. 2645–2657, 2015.

[9] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44(6), pp.2743–2760, 1998.

[10] F. Bonchi, and C. Lucchese, "Pushing tougher constraints in frequent pattern mining," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 114–124, 2005.

[11] M. A. Bhuiyanm, and M. Al Hasan, "An iterative MapReduce based frequent subgraph mining algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27(3), pp. 608–620, 2015.

[12] J. H. Chang, and W. S. Lee, "Finding recent frequent itemsets adaptively over online data streams," *ACM SIGKDD International Conference Knowledge Discovery and Data Mining*, pp. 487–492, 2003.

[13] H. Cheng, X. Yan, and J. Han, "IncSpan: incremental mining of sequential patterns in large database," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 527–532, 2004.

[14] J. H. Chang, and W. S. Lee, "Efficient mining method for retrieving sequential patterns over online data streams," *Journal of Information Science*, vol. 31(5), pp. 420–432, 2005.

[15] L. Chang, T. Wang, D. Yang, and H. Luan, "Seqstream: Mining closed sequential patterns over stream sliding windows," *IEEE International Conference on Data Mining*, pp. 83–92, 2008.

[16] C. H. Chen, A. F. Li, and Y. C. Lee, "Actionable high-coherent-utility fuzzy itemset mining," *Soft Computing*, vol. 18(12), pp. 2413–2424, 2014.

[17] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh, "iSAX 2.0: Indexing and mining one billion time series," *IEEE International Conference on Data Mining*, pp. 58–67, 2010.

[18] J. H. Chang, "Mining weighted sequential patterns in a sequence database with a time-interval weight," *Knowledge-Based Systems*, vol. 24(1), pp. 1–9, 2011.

[19] R. Campisano, F. Porto, E. Pacitti, F. Masseglia, and E. Ogasawara, "Spatial sequential pattern mining for seismic data," *The Brazilian Symposium on Databases*, pp. 241–246, 2016.

[20] L. Cao, X. Dong, and Z. Zheng, "e-NSP: Efficient negative sequential pattern mining," *Artificial Intelligence*, vol. 235, pp. 156–182, 2016.

[21] C. Fiot, A. Laurent, and M. Teisseire, "From crispness to fuzziness: Three algorithms for soft sequential pattern mining," *IEEE Transactions on Fuzzy Systems*, vol. 15(6), pp. 1263–1277, 2007.

[22] P. Fournier-Viger, R. Nkambou, and E. Mephu Nguifo, "A Knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems," *The Mexican International Conference on Artificial Intelligence*, pp. 765–778, 2008.

[23] T. C. Fu, "A review on time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 24(1), pp. 164–181, 2011.

[24] P. Fournier-Viger, and V. S. Tseng, "Mining top-k sequential rules, *The International Conference on Advanced Data Mining and Applications*, pp. 180–194, 2011.

[25] P. Fournier-Viger, T. Gueniche, and V. S. Tseng, "Using partially-ordered sequential rules to generate more accurate sequence prediction, *The International Conference on Advanced Data Mining and Applications*, pp. 431–442, 2012.

[26] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Mining top-k association rules, *The Canadian Conference on Artificial Intelligence*, pp.61–73, 2012.

[27] P. Fournier-Viger, A. Gomariz, T. Gueniche, E. Mwamikazi, and R. Thomas, "TKS: Efficient Mining of Top-K Sequential Patterns," *The International Conference on Advanced Data Mining and Applications*, pp. 109–120, 2013.

[28] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Mining Maximal Sequential Patterns without Candidate Maintenance," *The International Conference on Advanced Data Mining and Applications*, pp. 169–180, 2013.

[29] P. Fournier-Viger, A. Gomariz, T. Gueniche, E. Mwamikazi, and R. Thomas, "TKS: Efficient mining of top-k sequential patterns," *The International Conference on Advanced Data Mining and Applications*, pp. 109–120, 2013.

[30] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 40–52, 2014.

[31] P. Fournier-Viger, C.-W. Wu, A. Gomariz, and V. S. Tseng, "VMSP: Efficient vertical mining of maximal sequential patterns," *The Canadian Conference on Artificial Intelligence*, pp. 83–94, 2014.

[32] P. Fournier-Viger, A. Gomariz, M. Sebek, M. Hlosta, "VGEN: fast vertical mining of sequential generator patterns," *The International Conference on Data Warehousing and Knowledge Discovery*, pp. 476–488, 2014.

[33] P. Fournier-Viger, C. W. Wu, V. S. Tseng, "Novel concise representations of high utility itemsets using generator patterns," *The Internatioanl Conference on Advanced Data Mining and Applications*, pp. 30–43, 2014.

[34] P. Fournier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, "ERMiner: sequential rule mining using equivalence classes," *The International Symposium on Intelligent Data Analysis*, pp. 108–119, 2014.

[35] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, V. S. Tseng, "SPMF: A java open-source pattern mining library," *Journal of Machine Learning Research*, vol. 15, pp. 3389–3393, 2014.

[36] P. Fournier-Viger, C. W. Wu, S. Zida, V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," *The International Symptom on Methodologies for Intelligent Systems*, pp. 83–92, 2014.

[37] M. Fabregue, A. Braud, S. Bringay, F. Le Ber, and M. Teisseire, "Mining closed partially ordered patterns, a new optimized algorithm," *Knowledge-Based Systems*, vol. 79, pp. 68–79, 2015.

[38] P. Fournier-Viger, C. W. Wu, V. S. Tseng, L. Cao, R. Nkambou, "Mining partially-ordered sequential rules common to multiple sequences," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27(8), pp. 2203–2216, 2015.

[39] F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba, "CloFAST: closed sequential pattern mining using sparse and vertical id-lists," *Knowledge and Information Systems*, vol. 48(2), pp. 1–35, 2015.

[40] P. Fournier-Viger, C. W. Lin, A. Gomariz, A. Soltani, Z. Deng, H. T. Lam, "The SPMF open-source data mining library version 2," *The European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 36-40, 2016.

[41] P. Fournier-Viger, C. W. Lin, Q. H. Duong, and T. L. Dam, "PHM: Mining periodic high-utility itemsets," *The Industrial Conference on Data Mining*, pp. 64–79, 2016.

[42] M. N. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential pattern mining with regular expression constraints," *The International Conference on Very Large Databases*, pp. 223–234, 1999.

[43] B. Goethals, "Survey on frequent pattern mining," *University of Helsinki*, 2003.

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

[45] R. A. Garcia-Hernandez, J. F. Martanez-Trinidad, and J. A. Carrasco-Ochoa, "A new algorithm for fast discovery of maximal sequential patterns in a document collection," *The International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 514–523, 2006.

[46] C. Gao, J. Wang, Y. He, and L. Zhou, "Efficient mining of frequent sequence generators," *The International Conference on the World Wide Web*, pp. 1051–1052, 2008.

[47] K. Gouda, M. Hassaan, and M. J. Zaki, "Prism: An effective approach for frequent sequence mining via prime-block encoding," *Journal of Computer and System Sciences*, vol. 76(1), pp. 88–102, 2010.

[48] A. Gomariz, M. Campos, R. Marin, and B. Goethals, "ClaSP: An efficient algorithm for mining frequent closed sequences," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 50–61, 2013.

[49] J. Ge, Y. Xia, and J. Wang, "Towards efficient sequential pattern mining in temporal uncertain databases, *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 268–279, 2015.

[50] J. Ge, Y. Xia, and J. Wang, "Mining uncertain sequential patterns in iterative mapReduce," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 243–254, 2015.

[51] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. C. Hsu, "FreeSpan: frequent pattern-projected sequential pattern mining," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 355–359, 2000.

[52] T. P. Hong, K. Y. Lin, and S. L. Wang, "Mining fuzzy sequential patterns from multiple-items transactions," *The IFSA World Congress and the NAFIPS International Conference*, pp. 1317–1321, 2001.

[53] J. Han, J. Pei, Y. Ying, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8(1), pp. 53–87, 2004.

[54] J. Ho, L. Lukov, and S. Chawla, "Sequential pattern mining with constraints on large protein databases," *The International Conference on Management of Data*, pp. 89–100, 2005.

[55] C. C. Ho, H. F'. Li, F'. F. Kuo,and S. Y. Lee, "Incremental mining of sequential patterns over a stream sliding window," *IEEE International Conference on Data Mining Workshops*, pp. 677–681, 2006.

[56] K. Y. Huang, C. H. Chang, J. H. Tung, and C. T. Ho, "COBRA: closed sequential pattern mining using bi-phase reduction approach," *The International Conference on Data Warehousing and Knowledge Discovery*, pp. 280–291, 2006.

[57] S. C. Hsueh, M. Y. Lin, and C. L. Chen, "Mining negative sequential patterns for e-commerce recommendations," *IEEE Asia-Pacific Services Computing Conference*, pp. 1213–1218, 2008.

[58] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*, Amsterdam:Elsevier, 2011.

[59] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," *The Knowledge Engineering Review*, vol. 28(1), pp. 75–105, 2013.

[60] R. U. Kiran, and P. K. Reddy, "Mining rare periodic-frequent patterns using multiple minimum supports," *The International Conference on Management of Data*, 2009.

[61] R. U. Kiran, M. Kitsuregawa, and P. K. Reddy, "Efficient discovery of periodic-frequent patterns in very large databases," *Journal of Systems and Software*, vol. 112, pp. 110–121, 2016.

[62] M. Y. Lin, and S. Y. Lee, "Improving the efficiency of interactive sequential pattern mining by incremental pattern discovery," *The International Conference on System Sciences*, pp. 68–76, 2002.

[63] S. Lu, and C. Li, "AprioriAdjust: An efficient algorithm for discovering the maximum sequential patterns," *The International Workshop on Knowledge Grid and Grid Intelligence*, 2004.

[64] C. Luo, and S. Chung, "Efficient mining of maximal sequential patterns using multiple samples," *SIAM International Conference on Data Mining*, pp. 415–426, 2005.

[65] C. Lucchese, S. Orlando, and R. Perego, “Fast and memory efficient mining of frequent closed itemsets,” *IEEE Transactions on Knowledge Discovery and Data Engineering,* vol. 18(1), pp. 21–36, 2006.

[66] J. Lin, E. Keogh, L. Wei, and S. Lonardi, “Experiencing SAX: a novel symbolic representation of time series,” *Data Mining and Knowledge Discovery*, vol. 15(2), pp. 107–144, 2007.

[67] N. P. Lin, W.-H. Hao, H.-J. Chen, H.-E. Chueh, and C.-I. Chang, “Fast mining maximal sequential patterns,” *The International Conference on Simulation, Modeling and Optimization*, pp. 405-408, 2007.

[68] P. Lenca, B. Vaillant, P. Meyer, and S. Lallich, “Association rule interestingness measures: Experimental and theoretical studies,” *The Quality Measures in Data Mining Workshop*, pp. 51–76, 2007.

[69] Y. S. Lee, and S. J. Yen, “Incremental and interactive mining of web traversal patterns,” *Information Sciences*, vol. 178(2), pp. 287–306, 2008.

[70] D. Lo, S. C. Khoo, and J. Li, “Mining and ranking generators of sequential patterns,” *SIAM International Conference on Data Mining*, pp. 553–564, 2008.

[71] M. Liu, J. Qu, “Mining high utility itemsets without candidate generation,” *ACM International Conference Information and Knowledge Management*, pp.55–64, 2012.

[72] G. C. Lan, T. P. Hong, V. S. Tseng, and S. L. Wang, “Applying the maximum utility measure in high utility sequential pattern mining,” *Expert Systems with Applications*, vol. 41(11), pp. 5071–5081, 2014.

[73] J. C. W. Lin, L. Tin, P. Fournier-Viger, and T. P. Hong, “A fast Algorithm for mining fuzzy frequent itemsets,” *Journal of Intelligent and Fuzzy Systems*, vol. 9(6), pp.2373–2379, 2015.

[74] J. C. W. Lin, T. P. Hong, W. Gan, H. Y. Chen, and S. T. Li, “Incrementally updating the discovered sequential patterns based on pre-large concept,” *Intelligent Data Analysis*, vol. 19(5), pp. 1071–1089, 2015.

[75] J. C. W. Lin, T. P. Hong, and G. C. Lan, “Updating the sequential patterns in dynamic databases for customer sequences deletion,” *Journal of Internet Technology*, vol. 16(3), pp. 369–377, 2015.

[76] J. C. W. Lin, W. Gan, P. Fournier-Viger, and T. P. Hong, “Maintenance sequential patterns for sequence modification in dynamic databases,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 26(8), pp. 1285–1313, 2016.

[77] H. Mannila, H. Toivonen, A. I. Verkamo, “Discovery of frequent episodes in event sequences,” *Data Mining and Knowledge Discovery*, vol. 1(3), pp. 259–289, 1997.

[78] F. Masseglia, P. Poncelet, and M. Teisseire, “Incremental mining of sequential patterns in large databases,” *Data and Knowledge Engineering*, vol. 46(1), pp. 97–121, 2003.

[79] N. R. Mabroukeh, and C. I. Ezeife, “A taxonomy of sequential pattern mining algorithms,” *ACM Computing Surveys*, vol. 43(1), 2010.

[80] M. Muzammal, and R. Raman, “On probabilistic models for uncertain sequential pattern mining,” *The International Conference on Advanced Data Mining and Applications*, pp. 60–72, 2010.

[81] M. Muzammal, and R. Raman, “Mining sequential patterns from probabilistic databases,” *Knowledge and Information Systems*, vol. 44(2), pp. 325–358, 2015.

[82] S. N. Nguyen, X. Sun, M. E. Orlowska, “Improvements of IncSpan: Incremental mining of sequential patterns in large database,” *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 442–451, 2005.

[83] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules,” *The International Conference on Database Theory*, pp. 398–416, 1999.

[84] S. Parthasarathy, M. J. Zaki, M. Ogihara, and S. Dwarkadas, “Incremental and interactive sequence mining,” *The International Conference on Information and Knowledge Management*, pp. 251–258, 1999.

[85] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal, “Multi-dimensional sequential pattern mining,” *The International Conference on Information and Knowledge Management*, pp. 81–88, 2001.

[86] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang, “H-mine: Hyper-structure mining of frequent patterns in large databases,” *IEEE International Conference on Data Mining*, pp. 441–448, 2001.

[87] J. Pei, J. Han, and L. V. Lakshmanan, “Mining frequent itemsets with convertible constraints,” *The International Conference on Data Engineering*, pp.433–442, 2001.

[88] J. Pei, J. Han, and L. V. Lakshmanan, “Pushing convertible constraints in frequent itemset mining,” *Data Mining and Knowledge Discovery*, vol. 8(3), pp. 227–252, 2004.

[89] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on knowledge and data engineering*, vol. 16(11), pp. 1424–1440, 2004.

[90] J. Pei, H. Wang, J. Liu, K. Wang, J. Wang, and P. S. Yu, "Discovering frequent closed partial orders from strings," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18(11), pp. 1467–1481, 2006.

[91] J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods," *Journal of Intelligent Information Systems*, vol. 28(2), pp. 133–60, 2007.

[92] T. T. Pham, J. Luo, T.-P. Hong, and B. Vo, "MSGPs: a novel algorithm for mining sequential generator patterns," *The International Conference on Computational Collective Intelligence*, pp. 393–401, 2012.

[93] Y. W. T. Pramono, "Anomaly-based Intrusion Detection and Prevention System on Website Usage using Rule-Growth Sequential Pattern Analysis," *The International Conference on Advanced Informatics, Concept Theory and Applications*, pp. 203–208, 2014.

[94] J. M. Pokou, P. Fournier-Viger, and C. Moghrabi, "Authorship attribution using small sets of frequent part-of-speech skip-grams," *The International Florida Artificial Intelligence Research Society Conference*, pp. 86–91, 2016.

[95] M. N. Quang, T. Dinh, U. Huynh, and B. Le, "MHHUSP: An integrated algorithm for mining and hiding high utility sequential patterns," *The International Conference on Knowledge and Systems Engineering*, pp. 13–18, 2016.

[96] C. Raissi, P. Poncelet, and M. Teisseire, "SPEED: mining maxirnal sequential patterns over data streams," *IEEE Conference Intelligent Systems*, pp. 546–552, 2006.

[97] J. D. Ren, J. Yang, and Y. Li, "Mining weighted closed sequential patterns in large databases," *The International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 640–644, 2008.

[98] R. Srikant, and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," *The International Conference on Extending Database Technology*, pp. 1–17, 1996.

[99] P. Songram, and V. Boonjing, "Closed multidimensional sequential pattern mining," *International Journal of Knowledge Management Studies*, vol. 2(4), pp. 460–479, 2008.

[100] E. Salvemini, F. Fumarola, D. Malerba, and J. Han, "Fast sequence mining based on sparse id-lists," *The International Symposium on Methodologies for Intelligent Systems*, pp. 316–325, 2011.

[101] A. Soulet, F. Rioult, "Efficiently depth-first minimal pattern mining," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 28–39, 2014.

[102] S. J. Shin, D. S. Lee, and W. S. Lee, "CP-tree: An adaptive synopsis structure for compressing frequent itemsets over online data streams," *Information Sciences*, vol. 278(10), pp. 559-576, 2014.

[103] L. Szathmary, P. Valtchev, A. Napoli, R. Godin, A. Boc, and V. Makarenkov. "A fast compound algorithm for mining generators, closed itemsets, and computing links between equivalence classes," *Annals of Mathematics and Artificial Intelligence*, vol. 70(1-2), pp. 81–105, 2014.

[104] D. Schweizer, M. Zehnder, H. Wache, H. F. Witschel, D. Zanatta, and M. Rodriguez, "Using consumer behavior data to reduce energy consumption in smart homes: Applying machine learning to save energy without lowering comfort of inhabitants," *IEEE International Conference on Machine Learning and Applications*, pp. 1123–1129, 2015.

[105] S. K. Tanbeer, C. F. Ahmed, B. S. Jeong, and Y. K. Lee, "Discovering periodic-frequent patterns in transactional databases," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 242–253, 2009.

[106] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets," *IEEE International Conference on Data Mining Workshop on Frequent Itemset Mining Implementations*, 2004.

[107] B. Vo, T. P. Hong, and B. Le, "DBV-Miner: A Dynamic Bit-Vector approach for fast mining frequent closed itemsets," *Expert Systems with Applications*, vol. 39(8), pp. 7196–206, 2012.

[108] J. Wang, J. Han, and C. Li, "Frequent closed sequence mining without candidate maintenance," *IEEE Transactions on Knowledge Data Engineering*, vol. 19(8), pp. 1042–1056, 2007.

[109] Y. Xifeng, H. Jiawei, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Data Base," *SIAM International Conference on Data Mining*, pp. 166–177, 2003.

[110] X. Yan, and J. Han, "gSpan: Graph-based substructure pattern mining," *IEEE International Conference Data Mining*, pp.721–724, 2002.

[111] Z. Yang, and M. Kitsuregawa, "LAPIN-SPAM: An improved algorithm for mining sequential pattern," *The International Conference on Data Engineering Workshops*, pp. 1222–1222, 2005.

[112] U. Yun, and J. J. Leggett, "WSpan: Weighted Sequential pattern mining in large sequence databases," *The International IEEE Conference Intelligent Systems*, pp. 512–517, 2006.

[113] H. Yu, H. Yamana, "Generalized sequential pattern mining with item intervals," *Journal of Computers*, vol. 1(3), pp. 51–60, 2006.

[114] S. Yi, T. Zhao, Y. Zhang, S. Ma, and Z. Che, "An effective algorithm for mining sequential generators," *Procedia Engineering*, vol. 15, pp. 3653–3657, 2011.

[115] J. Yin, Z. Zheng, and L. Cao, "USpan: an efficient algorithm for mining high utility sequential patterns," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 660–668, 2012.

[116] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12(3), pp. 372–390, 2000.

[117] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42(1-2), pp. 31–60, 2001.

[118] Z. Zheng, Y. Zhao, Z. Zuo, and L. Cao, "Negative-GSP: an efficient method for mining negative sequential patterns," *The Australasian Data Mining Conference*, pp. 63–67, 2009.

[119] Z. Zheng, Y. Zhao, Z. Zuo, and L. Cao, "An efficient GA-based algorithm for mining negative sequential patterns," *The Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 262–273, 2010.

[120] M. J. Zaki, and C. J. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," *SIAM International Conference on Data Mining*, pp.457–473, 2012.

[121] Z. Zhao, D. Yan, and W. Ng, "Mining probabilistically frequent sequential patterns in large uncertain databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26(5), pp. 1171–1184, 2014.

[122] A. Zimmermann, "Understanding episode mining techniques: Benchmarking on diverse, realistic, artificial data," *Intelligent Data Analysis*, vol. 18(5), pp. 761–791, 2014.

[123] J. Zhang, Y. Wang, and D. Yang, "CCSpan: Mining closed contiguous sequential patterns," *Knowledge-Based Systems*, vol. 89, pp.1–13, 2015.

[124] S. Ziebarth, I. A. Chounta, and H. U. Hoppe, "Resource access patterns in exam preparation activities," *The European Conference on Technology Enhanced Learning*, pp. 497–502, 2015.

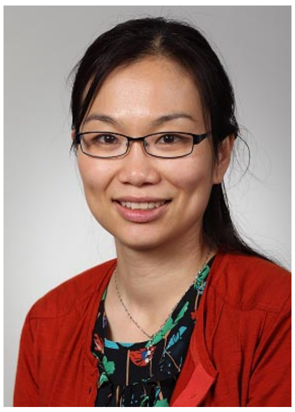
[125] S. Zida, P. Fournier-Viger, C. W. Wu, J. C. Lin, and V. S. Tseng, "Efficient mining of high-utility sequential rules," *The International Conference on Machine Learning and Data Mining*, pp.157–171, 2015.

[126] S. Zida, P. Fournier-Viger, J. C. W. Lin, C. W. Wu, and V. S. Tseng, "EFIM: A highly efficient algorithm for high-utility itemset mining," *The Mexican International Conference Artificial Intelligence*, pp. 530–546, 2015.

Philippe Fournier-Viger (Ph.D.) is a full professor and Youth 1000 scholar at the Harbin Institute of Technology Shenzhen Grad. School, China. He received a Ph.D. in Computer Science at the University of Quebec in Montreal (2010). He has published more than 130 research papers in refereed international conferences and journals, which have received more than 1,100 citations. His research interests include data mining, pattern mining, sequence analysis and prediction, text mining, e-learning, and social network mining. He is the founder of the popular SPMF open-source data mining library, which has been cited in more than 400 research papers since 2010. He is also editor-in-chief of the Data Mining and Pattern Recognition journal.

Jerry Chun-Wei Lin received his Ph.D. in Computer Science and Information Engineering at the National Cheng Kung University, Tainan, Taiwan in 2010. He is now working as the associate professor at the Harbin Institute of Technology, Shenzhen, China. He has published more than 200 research papers in refereed international conferences and journals, which have received more than 1000 citations. His research interests include data mining, privacy-preserving and security, big data analytics, and social network. He is the co-leader of the popular SPMF open-source data mining library and also the editor-in-chief of the Data Mining and Pattern Recognition (DSPR) journal.

Rage Uday Kiran (Ph.D.) is working as a Specially Appointed Research Assistant Professor at the University of Tokyo, Tokyo, Japan. He recieved his PhD degree in computer science from International Institute of Information Technology, Hyderabad, Telangana, India. His current research interests include data mining, ICTs for agriculture and recommender systems. He has published papers in International Conference on Extending Database Technology (EDBT), The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Database Systems for Advanced Applications (DASFAA), International Conference on Database and Expert Systems Applications (DEXA), International Conference on Scientific and Statistical Database Management (SSDBM), International Journal of Computational Science and Engineering (IJCSE), Journal of Intelligent Information Systems (JIIS), and Journal of Systems and Software (JSS).



Yun Sing Koh (Ph.D.) is currently a senior lecturer at the Department of Computer Science, The University of Auckland, New Zealand. She received her Ph.D. at the Department of Computer Science, University of Otago, New Zealand. Her research interest is in the area of data mining and machine learning, specifically data stream mining and pattern mining.

Rincy Thomas (M. Tech)(M. Tech) is an associate professor at SCT, Bhopal (M.P) India. He received his M.Tech from the Sagar Institute of Research & Technology Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India.