A Survey of Parallel Sequential Pattern Mining

Preprint · May 2018

CITATIONS
0

READS
717

5 authors, including:

Wensheng Gan
Jinan University (Guangzhou, China)
149 PUBLICATIONS **2,219** CITATIONS
SEE PROFILE

Chun-Wei Jerry Lin
Høgskulen på Vestlandet
607 PUBLICATIONS **9,238** CITATIONS
SEE PROFILE

Philippe Fournier Viger
Shenzhen University
383 PUBLICATIONS **8,887** CITATIONS
SEE PROFILE

Philip S. Yu
University of Illinois at Chicago
1,539 PUBLICATIONS **89,404** CITATIONS
SEE PROFILE

A Survey of Parallel Sequential Pattern Mining

Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger,
Han-Chieh Chao, *Senior Member, IEEE* and Philip S. Yu, *Fellow, IEEE*

Abstract—With the growing popularity of resource sharing and shared resources, large volumes of complex data of different types are collected automatically. Traditional data mining algorithms generally have problems and challenges including huge memory cost, low processing speed, and inadequate hard disk space. For sequential pattern mining (SPM), it is used in a wide variety of real-life applications. However, it is more complex and challenging than frequent itemset mining, and also suffers from the above challenges when handling the large-scale data. To solve these problems, mining sequential patterns in a parallel computing environment has emerged as an important issue with many applications. In this paper, an in-depth survey of the current status of parallel sequential pattern mining (PSPM) is investigated and provided, including detailed categorization of traditional serial SPM approaches, and state of the art parallel SPM. We review the related work of PSPM in detail, including partition-based algorithms for PSPM, Apriori-based PSPM, pattern growth based PSPM, and hybrid algorithms for PSPM, and provide deep description (i.e., characteristics, advantages, and disadvantages) of each parallel approach of PSPM. Some advanced topics for PSPM and the related open-source software are further reviewed in details. Finally, we summarize some challenges and opportunities of PSPM in the big data era.

Index Terms—Data mining, parallelism, sequential pattern, big data, efficiency.

1 Introduction

With the rapid development of information technology and data collection, Knowledge Discovery in Databases (KDD), which is also called data mining provides a powerful capability to discover meaningful and useful information from different types of complex data [1], [2], [3], [4], [5]. KDD has numerous real-life applications and is crucial to some of the most fundamental tasks such as frequent itemset and association rule mining [3], [4], [6], sequential pattern mining [5], [7], [8], clustering [9], [10], classification [11], outline detection [12].

Most traditional data mining algorithms are designed to run on a single computer (node), and are thus called single-node techniques. They can discover various kinds of patterns from various types of databases. At the same time, in recent decades, data mining has been studied extensively and applied widely [1], [3], [4], [9], [13], [14], [15]. These techniques perform well on small datasets, however, due to the limited memory capacity and computation capability of a single node, these data mining methods become inefficient over big data. The memory requirements for handling the complete set of desired results increase quickly, and the computational cost can be expensive on a single machine. All aforementioned methods are serialized. When handling large-scale data, these methods are fundamentally inappropriate due to many reasons, including the huge amounts of data, infeasibility of bandwidth limitation, as well as the fact that larger inputs demands parallel processing, and privacy concerns. Unfortunately, parallelization of the mining process is a difficult task. It is an important issue to develop more adaptable and flexible mining frameworks.

Parallel data mining (PDM) [16], [17] is a type of computing architecture in which several processors execute or process an application. Research and development work in the area of parallel data mining concerns the study and definition of parallel mining architectures, methods, and tools for the extraction of novel, useful, and implicit patterns from databases using a high-performance architecture. In some cases, PDM distributes the mining computation w.r.t. multicore over more than one node. When data mining tools are implemented on high-performance parallel computers, they can analyze massive databases in parallel processing within a reasonable time. In general, parallel computation allows for solving larger problems and executing applications that are parallel and distributed in nature. Parallel computing is becoming increasingly common and used to accelerate processing of the massive amount of data. So far, some parallelized Apriori-like algorithms have been implemented with the MapReduce framework [18] and achieved certain speedup compared with single-node methods. However, some previous studies [18], [19], [20] show that the MapReduce framework is not suitable for frequent itemset mining algorithm like the Apriori algorithm with intensive iterated computation.

In the field of data mining, pattern mining has become an important task for a wide range of real-world applications. Pattern mining consists of discovering interesting, useful, and unexpected patterns in databases. This field of research has emerged in the 1990s with the Apriori algorithm [1] which was proposed by Agrawal and Srikant. It is designed for finding frequent itemsets and then extracting the association rules. Note that frequent itemsets are the groups of items (symbols) frequently appearing together in a database of customer transactions. For example, the pattern/products $\{bread, wine, cheese\}$ can be used to find the shopping behavior of customers for market basket analysis.

- *W. Gan is with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China, and the Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA. E-mail: wsgan001@gmail.com. And corresponding author is this article is J.C.W. Lin, with the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China. E-mail: jerrylin@ieee.org.*

TABLE 1
An original sequence database w.r.t. shopping behavior.

tid	time	Customer ID	Event (products)
t_1	03-06-2017 10:05:30	C_1	*milk, bread*
t_2	03-06-2017 10:09:12	C_2	*oatmeal*
t_3	03-06-2017 10:21:45	C_3	*milk, bread, butter*
t_4	03-06-2017 11:40:00	C_1	*milk, cereal, cheese*
t_5	03-06-2017 12:55:30	C_3	*cereal, oatmeal*
t_6	03-06-2017 14:38:58	C_2	*bread, milk, cheese, butter, cereal*
...	...	...	...
t_{10}	05-06-2017 15:30:00	C_1	*bread, oatmeal, butter*

Some pattern mining techniques, such as frequent itemset mining (FIM) [1], [4] and association rule mining (ARM) [1], are aimed at analyzing data, where the sequential ordering of events is not taken into account. However, the sequence-based database which contains the embedded time-stamp information of event is commonly seen in many real-world applications. A sequence in a sequence database is an ordered list of items, and sequence is everywhere in our daily life. Typical examples include consumers' shopping behavior, Web access logs, DNA sequences in bioinformatics, and so on. We illustrate the sequential data with one case of market basket analysis in detail below. For example, Table 1 is a simple retail store's database which contains customers' shopping records, including transaction ID (tid), occur time, customer ID, and event (w.r.t. purchase products), and so on. For each customer, his/her total purchase behavior can be considered as a sequence consists of some events which happened at different time. As shown in Table 2 which is translated from Table 1, the customer C_1 w.r.t. sequence ID (SID = 1) has three records (t_1, t_4, and t_{10}) that occurred sequentially.

TABLE 2
A translated sequence database from Table 1.

SID	tid	Event (products)
1	t_1	*milk, bread*
1	t_4	*milk, cereal, cheese*
1	t_{10}	*bread, oatmeal, butter*
2	t_2	*oatmeal*
2	t_6	*bread, milk, cheese, butter, cereal*
3	t_3	*milk, bread, butter*
3	t_5	*cereal, oatmeal*

To address this issue, another concept of pattern mining named sequential pattern mining (SPM) was first introduced by Agrawal and Srikant in 1995 [2]. The goal of SPM aims at discovering and analyzing statistically relevant subsequences from sequences of events or items with time constraint. More precisely, 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. Given a user-specified threshold, termed the minimum support (denoted *minsup*), a sequence is said to be a *frequent sequence* or a *sequential pattern* if it occurs more than *minsup* times in the processed database. For instance, consider the database of Table 2, and assume that the user sets *minsup* = 3 to find all subsequences appearing in at least three sequences. For example, the patterns $\{milk\}$ and $\{bread, butter\}$ are frequent and have a support of 4 and 3 sequences, respectively. As a fundamental task of pattern mining, sequential pattern mining is used in a wide variety of real-life applications, such as market basket analysis [2], Web mining [21], bioinformatics [22], classification [23], finding copy-paste bugs in large-scale software code [24].

In the past two decades, pattern mining (i.e., FIM, ARM and SPM) has been extensively studied and successfully applied in many fields [6], [8]. Meanwhile, to meet the demand of large-scale and high performance computing, as mentioned before, parallel data mining has received considerable attention over the past decades [16], [17], [25], [26], including parallel frequent itemset mining (PFIM) [19], [27], [28], parallel association rule mining (PARM) [20], [29], parallel sequential pattern mining (PSPM) [30], [31], parallel clustering [32], [33], and so on. Among them, the sequence-based task - PSPM is crucial in a wide range of real-word applications. For example, in Bioinformatics for DNA sequence analysis [22], it requires a truly parallel computing on massive large-scale DNA. On one hand, the serial sequential pattern mining is computationally intensive. Although a significant amount of developments have been reported, there is still much room for improvement in its parallel implementation. On the other hand, many applications are time-critical and involve huge volumes of sequential data. Such applications demand more mining power than serial algorithms can provide [34]. Thus, solutions of the sequential pattern mining task that scale are quite important. Up to now, the problem of parallel sequential pattern mining has attracted a lot of attention [30], [31]. Fig. 1 shows the number of published papers from 2007 to 2017 where *Group 1* denotes the search keywords of "Sequential Pattern Mining" / "Sequence Mining", and *Group 2* denotes the search keywords of "Parallel Sequential Pattern Mining" / "Parallel Sequence Mining" / MapReduce "sequential pattern". Fig. 1 outlines a rapid surge of interest in SPM and PSPM in recent years. These results can easily show the trend that mining sequential patterns in a parallel computing environment has emerged as an important issue.

Up to now, several related surveys of serial sequential pattern mining (abbreviated as SPM) have been previously studied [35], [36], [37]. Recently, Fournier-Viger et al. published an up-to-date survey of the current status of serial SPM [8]. There is, unfortunately, no survey of parallel sequential pattern mining methods yet. In 2014, Anastasiu et al. published an article on the big data frequent pattern mining [38], which describes some Serial and parallel

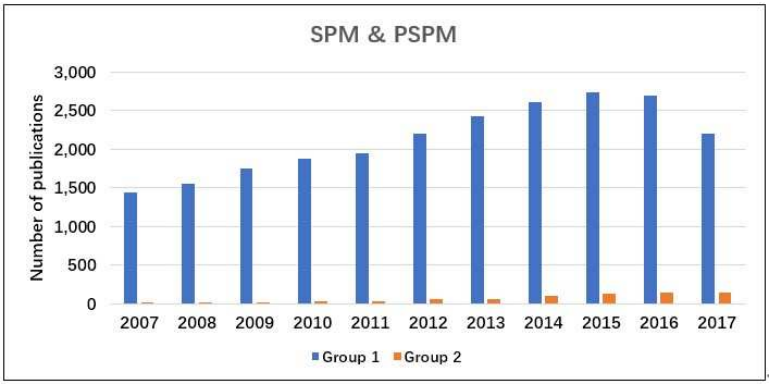

Fig. 1. Number of published papers that use "Parallel Sequential Pattern Mining" in subareas of data mining and big data. These publication statistics are obtained from *Google Scholar*; the search phrase of *Group 1* and *Group 2* is defined as the subfield named with the exact phrase "parallel sequential pattern" and at least one of parallel or sequential pattern/sequence appearing, e.g., "parallel sequence", "parallel sequential pattern".

approaches to scalable frequent itemset mining, sequence mining, and frequent graph mining. However, this article is not specific for parallel sequential pattern mining, only 12 algorithms for PSPM are presented in one section. Many concepts, technologies, big data platforms and tools, domain applications, and the state-of-the-art works of PSPM are not considered and reviewed. Yet, after more than ten years of theoretical development of big data, a significant number of new technologies and applications have appeared in this area. Therefore, we believe that now is a good time to summarize current status (i.e., the big data platforms and tools, the new technologies, application, and advanced topics) of PSPM.

The question then posed is: how can one best summarize the related studies in various types of parallel sequential pattern mining in parallel computing environments and make a general taxonomy of them? The methods summarized in this article not only fit for parallel computing [21], [39], [40], but are also good references for other related work, such as data mining [13], big data technologies [25], [26], [41], [42], [43], distributed systems [24], [44], and database management [45]. Thus, this paper aims at reviewing the current status of parallel sequential pattern mining (PSPM), and providing in-depth descriptions on a number of representative algorithms. The main contributions of this paper are described below.

- We review some related works on serial sequential pattern mining in several categories, including Apriori-based techniques for SPM, pattern growth techniques for SPM, algorithms for SPM with early-pruning, and constraint-based algorithms for SPM.
- We review the state-of-the-art works of parallel sequential pattern mining on distributed environments in recent years. This is a high level survey about parallel techniques for sequential pattern mining in several aspects, including partition-based algorithms for PSPM, Apriori-based PSPM, pattern growth based PSPM, and hybrid algorithms for PSPM. Not only the representative algorithms, but also the advances on latest progresses are reviewed comprehensively. We also point out the key idea, advantages and disadvantages of each PSPM approach.
- Some advanced topics for PSPM are reviewed, including PSPM with quantitative / weighted / utility, PSPM from uncertain data and stream data, hardware accelerator for PSPM (i.e., CPU and GPU approaches for parallel sequential pattern mining).
- We further review some well-known open-source software in the fields of serial SPM and parallel SPM, hope that these resources may reduce barriers for future research.
- Finally, some challenges and opportunities of parallel sequential pattern mining task for future research are briefly summarized and discussed.

The rest of this paper is organized as follows. Section 2 first introduces the key characteristics about some parallelism methods, then reviews some important features of parallel computing platforms and distributed systems, and respectively summarizes some important parallel computing platforms and tools. The definitions of major concepts used in literature are first introduced in Section 3. Besides, Section 3 also briefly provides the state-of-the-art research efforts SPM. Section 4 highlights and discusses the state-of-the-art research efforts on different approaches, under four taxonomies, for parallel sequential pattern mining. Some advanced topics for PSPM and the related open-source software are further reviewed in Section 5 and Section 6, respectively. Section 7 briefly summarizes some challenges and opportunities in parallel sequential pattern mining. Finally, some conclusions are briefly given in Section 8.

2 Parallel Computing Framework

2.1 Methods of Parallelism

Zaki has pointed several reasons for designing parallel algorithms to the data mining task [16], [46]. First, single processors cause the memory and CPU speed limitations, while multiple processors can parallelly reduce them. Second, large amounts of data are already available in parallel sub-databases or they are stored/distributed at multiple sites [16], [46]. Therefore, it is prohibitively expensive either to collect all data into one site, or to perform the serial mining process on a single computer. For these reasons, tools that can help in parallelization are urgently needed. According to the previous studies [16], [46], there are some well-known methods of parallelism: task parallelism (divide

and conquer strategy, task queue), data parallelism (record based and attribute based), and hybrid data/task parallelism. Characteristics about these parallelism methods are described below.

(1) Task parallelism[1] Task parallelism assigns portions of the search space to separate processors. Thus, different tasks running on the same data. There are two types of task parallel approaches: based on divide and conquer strategy and based on a task queue. The former divides the search space and assigns each partition to a specific processor, while the later dynamically assigns small portions of the search space to a processor whenever it becomes available.

(2) Data parallelism[2] Data parallelism distributes the dataset over the multiple available processors. The same task parallelly run on different data in distributed environment. In general, data-parallel approaches come in two flavors: record-based data parallelism and attribute-based data parallelism.

(3) Hybrid data/task parallelism It is a parallel pipeline of tasks, where each task might be data paralleled, and output of one task is the input to the next one. It means that each task can be run in parallel, throughput impacted by the longest-latency element in the pipeline.

Nowadays, parallel computing is the key to improve the performance of computer programs. Currently, data mining approaches to achieve parallelization and distribution can be classified in terms of five main components [16], [46]:

- Distributed versus shared memory systems;
- Data versus task parallelism;
- Static versus dynamic load balancing;
- Complete versus heuristic candidate generation;
- Horizontal versus vertical data layout.

2.2 Difference between Parallel Computing Platform and Distributed System

Unlike traditional centralized systems, a distributed system is defined as: one in which components of networked computers communicate and coordinate their actions only by passing messages [24], [44]. In other words, a distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system. According to [24], [44], there are two aspects of distributed systems: (1) independent computing elements and (2) single system w.r.t. middle-ware. There are some important features of distributed systems, including: 1) concurrency, multi-process and multi-threads concurrently execute and share resources; 2) no global clock, where program coordination depends upon on message passing; and 3) independent failure, wherein some processes failure cannot be known by other processes [24], [44]. There are many types of distributed systems, such as grids [47], [48], peer-to-peer (P2P) systems [49], ad-hoc networks [50], cloud computing systems [51], and online social network systems [52]. According to a study by Almasi et al., parallel computing[3] is a type of computation in which many calculations or the execution of processes are carried out simultaneously [53]. In general, large problems can often be divided into smaller ones, which can then be solved at the same time. In summary, there are three types of parallelism for parallel computing, including bit-level parallelism, instruction-level parallelism, and task parallelism.

Both distributed mining and parallel mining can speed up the data mining process. However, they have several differences. In 1999, Zaki pointed out that there are some differences between parallel and distributed association rule mining [16]. For achieving the parallelism of a traditional mining method, the main challenges include: i) synchronization minimization and communication minimization, ii) workload balancing, iii) finding good data layout and data decomposition, and iv) disk I/O minimization (which is especially important for association rule mining) [16]. In general, the parallel paradigm is hardware or software-distributed shared-memory systems. Thus, one parallel method can be distributed or shared-memory. In distributed data mining (DDM), the same or different mining algorithms are often applied to tackle local data. A DDM algorithm communicates among multiple process units, and then combines the local patterns into the desired global patterns. While a parallel data mining (PDM) algorithm applies the global parallel algorithm on the entire dataset to discover the desired knowledge, it is the same as that found by the traditional data mining algorithm. The accuracy or efficiency of DDM depends on data partitioning, task scheduling, and global synthesizing, and thus it is somewhat difficult to predict [54]. PDM accuracy may be more guaranteed than that of DDM.

2.3 Parallel Data Mining Platforms and Tools

In order to achieve high-performance data mining, some parallel data mining platforms and tools have been further developed in recent years. Many researchers provide different techniques to work on parallel or distributed environments like multi-core computing [55], grid computing [47], [48], graphics processing units (GPUs) [21], [39], cloud computing [51], Hadoop[4], etc. In recent years, the focus on computer engineering research shifted to exploit architecture advantages as much as possible, such as shared memory [56], FPGA [57], cluster architecture [53], or the massive parallelism of GPUs [58]. Some parallel data mining platforms and tools are first described below.

(1) Multi-core computing. A multi-core processor[5] includes multiple processing units (called "*cores*") on the same chip. This processor supports multi-core computing, and differs from a super-scalar processor. It can issue multiple instructions per clock cycle from multiple instruction streams [55].

(2) Grid computing. Generally speaking, grid computing[6] is a form of parallel computing, and different from conventional computing systems. It makes use of the collection of multiple distributed computer resources to fulfill a common goal. Each node in grid computers performs a different task/application.

1. http://en.wikipedia.org/wiki/Task_parallelism
2. http://en.wikipedia.org/wiki/Data_parallelism
3. https://en.wikipedia.org/wiki/Parallel_computing
4. http://hadoop.apache.org
5. https://en.wikipedia.org/wiki/Multi-core_processor
6. https://en.wikipedia.org/wiki/Grid_computing

(3) Reconfigurable computing with FPGA. A field-programmable gate array (FPGA) is, in essence, a computer chip that can rewire itself for a given task [57]. A new computing paradigm, reconfigurable computing, has received wide attention and is the focus of extensive research. Reconfigurable computing uses a FPGA as a co-processor to a general-purpose computer. Thus, high-performance computing can be achieved by using FPGA. Reconfigurable computing with FPGA has a good ability to combine efficiency with flexibility.

(4) Graphics processing units (GPUs)[7]. GPUs have attracted a lot of attention due to their cost-effectiveness and enormous power for massive data parallel computing. At present, a fairly recent trend is to enable general-purpose computing on graphics processing units. GPUs are co-processors that have been heavily optimized for computer graphics processing [21], [39]. Recently, some GPU-based tools are developed, such as a parallel graph exploration system on multi-core CPU and GPU [59], and a novel parallel algorithm on GPUs to accelerate pattern matching [60].

(5) MapReduce [18]. The MapReduce model was originally proposed by Google. It is a popular parallel programming model that not only simplifies the programming complexity of parallel or distributed computing, but also can achieve better processing and analytics for massive datasets. The MapReduce model [18] stores the data in *<key, value>* pair and then runs in rounds, which are composed of three consecutive phases: *map*, *shuffle*, and *reduce*. The input and output formats of MapReduce can be expressed as follows: i) **Mapper**: *<key input, value input>* to *list<key map, value map>*; ii) **Reducer**: *<key map, list(values)>* to *list<key reducer, value reducer>*. By using the MapReduce programming model, mining progress can be effectively enhanced. Moreover, I/O cost caused by scanning for massive and high dimensional datasets can be significantly reduced. Up to now, there are many implementations of MapReduce, and MapReduce has become a common powerful tool for parallel or distributed computing under various environments.

(6) Spark [61]. Spark is one of the distributed computing framework developed by Berkeley AMPLab. It is a well-known open-source cluster computing system that provides a flexible interface for programming entire clusters with implicit data parallelism and fault-tolerance. Due to the in-memory parallel execution model, all data will be loaded into memory in Spark. This fundamental feature means Spark can significantly speed up computation for iterative algorithms such as the Apriori algorithm and some other machine learning algorithms. In general, Spark [61] can be 1-2 orders of magnitude faster than MapReduce [18].

(7) PerfExplorer [62]. It is a computing framework for large-scale parallel data mining. It consists of two main components, the Client and the Server. PerfExplorer has a flexible interface and provides a common, reusable foundation for multiple data analysis, including clustering, dimensionality reduction, coefficient of correlation analysis, and comparative analysis [62].

7. https://en.wikipedia.org/wiki/Graphics_processing_unit

3 State-of-the-art Serial Sequential Pattern Mining

Most of the current parallel sequential pattern mining algorithms are extended from the traditional serial SPM algorithms. Therefore, in this section, some preliminaries and the problem statement related to frequent itemset mining (FIM) and sequential pattern mining (SPM) are first introduced. Then, some related works of sequential pattern mining are reviewed in several categories. Finally, the status of SPM is described and briefly summarized.

3.1 Frequent Itemset Mining VS. Sequential Pattern Mining

First, let us introduce some preliminaries and the problem statement related to frequent itemset mining (FIM) from transactional databases. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items, an *itemset* $X = \{i_1, i_2, \ldots, i_k\}$ with k items as a subset of I. The length or size of X is denoted as $|X|$, i.e., the number of items in X w.r.t. k. Given a transactional database $D = \{t_1, t_2, \ldots, t_m\}$, where each transaction $T_q \in D$ is generally identified by a transaction id (*TID*), and $|D|$ denotes the total number of transactions w.r.t. m. The support of X in D is denoted as $sup(X)$, it is the proportion of transactions that contain X, i.e., $sup(X) = |T_q|T_q \in D, X \subseteq T_q|/|D|$. An itemset is said to be a *frequent itemset* (*FI*) in a database if its support is no less than the user defined minimum support threshold ($minsup$). Therefore, the problem of frequent itemset mining is to discover all itemsets which have a support no less than the defined minimum support threshold, that is $sup(X) \geq minsup$ [1]. As the most important mining task for a wide range of real-world applications, frequent itemset mining (FIM) or association rule mining (ARM) has attracted a lot of attention [1], [3], [4], [6], [13], [14], [17].

In 1995, Srikant and Agrawal first extended the concept of the frequent itemset mining model to handle sequences [2]. They first introduced and formulated the sequential pattern mining (SPM) problem. Different from FIM, SPM aims at discovering frequent subsequences as interesting patterns in a sequence database which contains the embedded time-stamp information of event. According to the definitions from Mannila et al. [63], we have the associated notations and descriptions of SPM as below. Given an input sequence database $SDB = \{S_1, S_2, \ldots, S_j\}$, where *SID* is a unique sequence identifier (also called *sequence-id*) and S_k is an input sequence. Thus, *SDB* is a set of tuples (*SID*, S), and each sequence S has the following fields: *SID*, event-time, and the items present in the *event*. For example, in Table 1, consider the first sequence $<\{a, d\}, \{c\}, \{d, g\}, \{g\}, \{e\}>$, it represents five transactions made by a customer at a retail store. Each single letter represents an item (i.e., $\{a\}, \{c\}, \{d\}$, etc.), and items between curly brackets represent an itemset (i.e., $\{a, d\}$ and $\{d, g\}$). Simply speaking, a *sequence* is a list of temporally ordered itemsets (also called *events*).

A sequence $s_a = <A_1, A_2, \ldots, A_n>$ is called a *k-sequence* if it contains k items, or in other words if $k = |A_1| + |A_2| + \ldots + |A_n|$. For example, as shown in Table 3, the sequence $<\{a, d\}, \{c\}, \{d, g\}, \{g\}, \{e\}>$ is a 7-sequence. A sequence $S_\alpha = <\alpha_1, \alpha_2, \ldots, \alpha_n>$ is called a *sub-sequence* of another sequence $S_\beta = <\beta_1, \beta_2, \ldots, \beta_m>$ $(n < m)$, and

TABLE 3
A sequence database.

SID	Sequence
1	$<\{a,d\},\{c\},\{d,g\},\{g\},\{e\}>$
2	$<\{a\},\{d\},\{c,g\},\{e\}>$
3	$<\{a,b\},\{c\},\{f,g\},\{a,b,g\},\{e\}>$
4	$<\{b\},\{c\},\{d,f\}>$
5	$<\{a,b\},\{c\},\{d,f,g\},\{g\},\{e\}>$

S_β is called a *super-sequence* of S_α if there exists an integer $1 < i_1 <\ldots< i_n < m$ such that $\beta_1 \subseteq \beta_{i1},\ldots,\beta_{in} \subseteq \beta_{im}$, denoted as $S_\alpha \subseteq S_\beta$. A tuple (*SID*, S) is said to contain a sequence S_α if S is a *super-sequence* of S_α, denoted as $S_\alpha \subseteq S$. As shown in Table 3, the sequence $<\{a,d\},\{c\},\{g\}>$ is contained in sequence $<\{a,d\},\{c\},\{d,g\},\{g\},\{e\}>$, while the sequence $<\{a,d\},\{c,d\},\{g\}>$ is not. Especially, the sequence $<\{a,b\},\{c\}>$ and $<\{a,b\},\{d\},\{g\}>$ are called $sequence$-extensions of $<\{a,b\}>$, and $<\{a,b,d\}>$ is called $item$-extension of $<\{a,b\}>$. The support of a sequence S_α in a sequence database *SDB* is the number of tuples that contains S_α, and it is denoted as $sup(S_\alpha)$. The size of a sequence database *SDB* (denoted as $|SDB|$) corresponds to the total number of sequences (i.e., the number of customers).

In general, sequential pattern mining is more complex and challenging than frequent itemset mining. There are some important reasons for this. Firstly, due to the absence of time constraints in FIM not presenting in SPM, SPM has a potentially huge set of candidate sequences [64]. The total number of possible sequential patterns is much larger than that of FIM. If all sequences contain exactly one event, SPM is equal to FIM. For example, consider only 100 distinct items for the SPM problem, there are 100 sequences has their length as 1, while the number of sequences with length 2 is $100 \times 100 \times (100 \times 99/2)$ = 14,950, and the number of sequences with length 3 is $2^{100} - 1 \approx 10^{30}$. Secondly, there are further difficulties mining longer sequential patterns [64], such as DNA sequence analysis or stock sequence analytics/prediction.

3.2 State-of-the-art Serial Sequential Pattern Mining

Through 20 years of study and development, many techniques and approaches have been proposed for mining sequential patterns in a wide range of real-word applications [8], such as Web mining [21], classification [23], and mining motifs from biological sequences [22]. Some well-known serial algorithms for sequential pattern mining, such as AprioriAll [2], GSP [5], BIDE [65], CloSpan [66], FreeSpan [67], PrefixSpan [64], SPADE [68], etc., have been proposed. Since many parallel SPM algorithms are extended by the traditional serial SPM approaches, it is quite necessary for us to have an understanding of the systematic characteristics of traditional serial sequential pattern mining approaches. Thus, the state-of-the-art of serial SPM are reviewed below. Based on the different mining mechanisms, some well-known serial SPM approaches are divided into several categories, including Apriori-based techniques, pattern growth techniques, algorithms with early-pruning, and constraint-based algorithms. Details are summarized in Tables 4, 5, 6, and 7, respectively.

Apriori-Based Techniques. In 1995, Agrawal and Srikant first introduced an new approach named AprioriAll to discover the sequential patterns from a set of sequences whithin the embedded timestamp information [2]. The AprioriAll algorithm is based on Apriori and relies on the Apriori property. That is "all nonempty subsets of a frequent itemset must also be frequent" [1]. Therefore, AprioriAll holds the *downward closed property* (also called *anti-monotonic*), and utilizes this anti-monotonic to prune the search space. In general, two properties are often utilized in sequential pattern mining to speed up computation:

- ***Subsequence infrequency based pruning***: any super-sequence of an infrequent sequence is not frequent, thus it can be pruned from the set of candidates [2]. If S_a is a subsequence of S_b, then $sup(S_b)$ is at most as large as $sup(S_a)$. *Monotonicity*: If S_a is not frequent, then it is impossible that S_b is frequent. *Pruning*: If we know that S_a is not frequent, we do not have to evaluate any supersequence of S_a. For example, if sequence $<a>$ occurs only 10 times, then $<a,b>$ can occur at most 10 times.
- ***Supersequence frequency based pruning***: any subsequence of a frequent sequence is also frequent, thus it can be safely pruned from the set of candidates [77]. For example, if sequence $<a,b,d>$ is frequent and occurs 8 times, then any its subsequence ($<a>$, $<b>$, $<d>$, $<a,b>$, $<a,d>$ and $<b,b>$) is frequent and occurs at least 8 times.

Srikant then proposed GSP, which is similar to AprioriAll in execution process, but adopts several technologies including time constraints, sliding time windows, and taxonomies [5]. GSP uses a multiple-pass, candidate generation-and-test method to find sequential patterns. It adopts the Apriori property that all sub-patterns of a frequent pattern must be frequent. Thus, GSP can greatly improve performance over AprioriAll. PSP [69] resumes the general principles of GSP but it utilizes a different intermediary data structure, making it more efficient than GSP. AprioriAll [2] AprioriSome [2], DynamicSome [2] and GSP [5] all use the breadth-first search (BFS) to mine sequential patterns with a hierarchical structure. The SPADE algorithm, which uses equivalence classes to discover the sequential patterns, is an Apriori-based hybrid miner, it can be either breadth-first or depth-first [68]. SPADE exploits sequential patterns by utilizing a vertical id-list database format and a vertical lattice structure. SPAM is similar to SPADE, but SPAM uses bitmap representation and bitwise operations rather than regular and temporal joins. Yang et al. then proposed a novel algorithm LAPIN for SPM with last position induction (LAPIN-LCI and LAPIN-Suffix) [70]. They further developed the LAPIN-SPAM [71] algorithm by combining LAPIN and SPAM. Note that for SPAM, LAPIN, LAPIN-SPAM and SPADE, these algorithms use the property that the support of a super-patterns is always less than or equal to the support of its support patterns, it is different from the Apriori property used in GSP. Most of the above Apriori-based algorithms consist of three features.

- **Breadth-first search**: Apriori-based algorithms commonly use the breadth-first (w.r.t. level-wise) search

TABLE 4
Apriori-based serial algorithms for sequential pattern mining.

Name	Description	Year
Apriori (All, Some, Dynamic Some) [2]	The first algorithm for sequential pattern mining.	1995
GSP [5]	Generalized Sequential Patterns (Max/Min Gap, Window, and Taxonomies).	1996
PSP [69]	Retrieval optimizations and more efficient than GSP.	1998
SPADE [68]	Sequential PAttern Discovery using Equivalence classes.	2001
SPAM [23]	Sequential PAttern Mining with Bitmap representation.	2002
LAPIN [70]	SPM with LAst Position Induction, which is categorized into two classes, LAPIN-LCI and LAPIN-Suffix.	2004
LAPIN-SPAM [71]	LAst Position INduction Sequential PAttern Mining.	2005

TABLE 5
Pattern growth algorithms for sequential pattern mining.

Name	Description	Year
FreeSpan [67]	FREquEnt pattern-projected Sequential Pattern mining.	2000
WAP-mine [72]	SPM with suffix growth.	2000
PrefixSpan [64]	PREFIX-projected Sequential PAtterN mining.	2001
LPMiner [73]	Sequential pattern mining with Length decreasing suPport.	2001
SLPMiner [74]	Sequential pattern mining with Length decreasing suPport.	2002
FS-Miner [75]	SPM with suffix growth.	2004
LAPIN-Suffix [70]	SPM with suffix growth.	2004
PLWAP [76]	SPM with prefix growth.	2005

manner. They construct all k-sequences together in each k-th iteration while exploring the search space.

- **Generate-and-test**: This feature is introduced by Agrawal et al. in the Apriori algorithm [1]. It is used in the early sequential pattern mining algorithms, such as AprioriAll [2], AprioriSome [2], Dynamic Some [2].
- **Multiple database scans**: These algorithms need to scan the original database many times to determine whether a longer generated sequences is frequent. They suffer from the drawbacks of the candidate generation-and-test paradigm. In other words, they may generate a huge number of candidate sequences that do not appear in the input database and need to scan the original database many times.

Pattern-Growth Techniques. In 2000, a pattern-projected sequential pattern mining algorithm named FreeSpan was introduced by Han et al. [67]. By using a frequent item list (f-list) and S-Matrix, it obtains all frequent sequences based on so-called projected pattern growth. Pattern-growth SPM algorithms can avoid recursively scanning the database to find frequent patterns. The reason is that they only consider the patterns actually appearing in the database. To reduce time-consuming database scans, pattern-growth SPM algorithm introduces a novel concept called projected database [39], [67]. The projected database can significantly reduce the size of databases as frequent patterns are considered by the depth-first search.

The WAP-mine algorithm, which utilizes a WAP-tree, was proposed by Pei et al. [72]. In a WAP-tree, each node is assigned a binary code to determine which sequences are the suffix sequences of the last event, and then to find the next prefix for a discovered suffix. Thus, PLWAP does not have to reconstruct intermediate WAP-trees. Han et al. then further proposed the most representative algorithm PrefixSpan [64]. It tests only the prefix subsequences, and then projects their corresponding postfix subsequences into the projected sub-databases (A projected database is the set of suffixes w.r.t. a given prefix sequence). By recursively exploring only local frequent sequences, sequential patterns can be recursively grown in each projected subdatabase. Thus, PrefixSpan exhibits better performance than both GSP [5] and FreeSpan [67]. However, when dealing with large dense databases that have large itemsets, it is worse than that of SPADE [68]. As shown in Table 5, some other pattern growth algorithms for sequential pattern mining have been developed, such as LPMiner [73], SLPMiner [74], WAP-mine [72], and FS-Miner [75]. With consideration of length decreasing support, the LPMiner [73] and SLPMiner [74] algorithms are introduced. In 2005, Ezeife et al. proposed a pattern-growth and early-pruning hybrid method named PLWAP [76]. PLWAP utilizes a binary code assignment method to construct a preordered, linked, position-coded Web access pattern tree (WAP-tree for short). Compared to the WAP algorithm, PLWAP can significantly reduce runtime and provide a position code mechanism. Inspired by FP-tree [4] and the projection technique for quickly mining sequential patterns [39], a tree algorithm named FS-Miner (Frequent Sequence Miner) is developed [75]. FS-Miner resembles WAP-mine [72]. Moreover, it supports incremental mining and interactive mining [80].

Early-Pruning Techniques. Although the projection-based approaches (i.e., FreeSpan, PrefixSpan) can achieve a significant improvement over Apriori-based approaches, the projection mechanism still suffers from some drawbacks. The major cost of PrefixSpan is caused by constructing projected databases. Therefore, some hybrid algorithms with early-pruning strategy are developed, as shown in Table 6. Following in the footsteps of SPAM, a first-Horizontal-last-Vertical scanning database Sequential pattern Mining algorithm (named HVSM for short) [78] is developed using bitmap representation. Instead of using candidate generate-

TABLE 6
Algorithms for sequential pattern mining with early-pruning.

Name	Description	Year
HVSM [78]	Bitwise operation and position induction.	2005
LAPIN-SPAM [71]	Bitwise operation.	2005
PLWAP [76]	Position induction.	2005
LAPIN [70]	Position induction.	2007
DISC-all [22]	Position induction and prefix growth.	2007
UDDAG [79]	Up-Down Directed Acyclic Graph for sequential pattern mining.	2010

TABLE 7
Constraint-based algorithms for sequential pattern mining.

Name	Description	Constraint	Character	Year
CloSpan [66]	The first algorithm for mining closed sequential patterns.	• Closed	• Based on the GSP algorithm.	2003
BIDE [65]	Mining of frequent closed sequences by using BI-Directional Extension, which is a sequence closure checking scheme.	• Closed	• It executes some checking steps in the original database that avoid maintaining the sequence tree in memory.	2004
MSPX [81]	MSPX mines maximal sequential patterns with a bottom-up search and uses multiple samples.	• Maximal	• The sampling technique is used at each pass to distinguish potentially infrequent candidates.	2005
SQUIRE [82]	Sequential pattern mining with quantities, Apriori-QSP and PrefixSpan-QSP.	• Pattern with quantities	• Apriori-based and PrefixSpan-based.	2007
ClaSP [83]	Mining frequent closed sequential patterns by using several search space pruning methods together with a vertical database layout.	• Closed	• Inspired on the SPADE and CloSpan algorithms.	2013
MaxSP [56]	Maximal Sequential Pattern miner without candidate maintenance.	• Maximal	• It neither produces redundant candidates nor stores intermediate candidates in main memory.	2013
VMSP [84]	The first vertical mining algorithm for Vertical mining of Maximal Sequential Patterns.	• Maximal	• Uses a vertical representation to do a depth-first exploration of the search space.	2014
CloFAST [85]	A fast algorithm for mining closed sequential patterns using id-list.	• Closed	• It combines a new data representation of the dataset, based on sparse and vertical id-list.	2016

and-test or the tree projection, the LAPIN algorithm [70] uses an item-last-position list and a position set of prefix border. Based on the anti-monotone property which can prune infrequent sequences, Direct Sequence Comparison (DISC-all) uses other sequences of the same length to prune infrequent sequences [22]. The DISC-all respectively employs different orderings (lexicographical ordering and temporal ordering) to compare sequences of the same length. With the up-to-down directed acyclic graph, the UDDAG algorithm [79] uses the prefixes and suffixes to detect the pattern. It obtains faster pattern growth because of fewer levels of database projection compared to other approaches, and allows bi-directional pattern growth along both ends of detected patterns.

Constraint-based Techniques. Different from the traditional SPM approaches, the interesting issue of constraint-based SPM has been widely studied, including quantitative sequences, maximal sequential pattern, closed sequential patterns, sequential patterns with gap constraint, top-k sequential patterns, etc. Some constraint-based SPM algorithms are described in Table 7. In order to handle the condense representation of an explosive number of frequent sequential patterns, the issue of mining maximal sequential patterns is first developed. The MSPX mines maximal sequential patterns with a bottom-up search and uses multiple samples [81]. Unlike the traditional single-sample techniques used in FIM algorithms, MSPX uses multiple samples at each pass to distinguish potentially infrequent candidates. Then, an efficient algorithm called MaxSP (Maximal Sequential Pattern miner without candidate maintenance) was proposed [56]. The maximal representation may cause the information loss of support, thus another condense representation named closed pattern was introduced [66]. Up to now, some algorithms for mining closed sequential patterns have been proposed, such as CloSpan [66], BIDE [65], ClaSP [83], and CloFAST [85]. CloSpan [66] adopts the candidate maintenance-and-test method to prune the search space, and it may perform worse while database having long sequences or the support threshold is very low. A novel closure checking scheme, called bi-directional extension, is developed in BIDE [65]. BIDE mines closed sequential patterns without candidate maintenance, it is more efficient than CloSpan. Gomariz et al. then introduced the ClaSP algorithm by using several efficient search space pruning methods together with a vertical database layout [83]. Recently, a more fast algorithm called CloFAST was

proposed for mining closed sequential patterns using sparse and vertical id-lists [85]. CloFAST does not build pseudo-projected databases and does not need to scan them. It is more efficient than the previous approaches, including CloSpan, BIDE, and ClaSP.

Discussions. Each method of serial sequential pattern mining algorithm has advantages and disadvantages. Besides, there are also many different definitions of categories for SPM algorithms. A more comprehensive discussion of these methods has been given in [35], [36], [37], and an up-to-date survey of the current status of sequent pattern minng can be referred to [8].

4 State-of-the-art Parallel Sequential Pattern Mining

In this section, we first briefly overview the current status of parallel frequent itemset mining. We also cover some special categories of parallel sequential pattern mining algorithms, i.e., partition-based algorithms for PSPM, Apriori-based algorithms for PSPM, pattern growth algorithms for PSPM, and hybrid algorithms for PSPM. In most PSPM algorithms, they are hybrid inherently.

4.1 Parallel Frequent Itemset Mining

As mentioned previously, sequential pattern mining is highly related to frequent itemset mining, and SPM is more complicated than FIM. Since some developments of SPM are inspired by many technologies of FIM, it is needed to have an overview of the current development in parallel frequent itemset mining (PFIM). The problem of frequent itemset mining in parallel environments has been extensively studied so far, and a number of approaches have been explored to address this problem, such as PEAR [86], PPAR [86], PDM [87], ParEclat [29], PaMPa-HD [27], PHIKS [28], etc. In 1995, Mueller first proposed two parallel algorithms, Parallel Efficient Association Rules (PEAR) [86] and Parallel Partition Association Rules (PPAR) [86]. Cheung et al. also proposed an algorithm named Parallel Data Mining (PDM) to parallel mining of association rules [87], and the Fast Distributed Mining (FDM) algorithm for distributed databases [88]. An adaptation of the FP-Growth algorithm to MapReduce [18] called PFP (parallel FP-tree) is presented in [19]. PFP is a parallel form of the classical FP-Growth, it splits a large-scale mining task into independent and parallel tasks. By extending the vertical mining approach Eclat [89], the parallel-based Eclat (ParEclat) [29] and the distributed Eclat (Dist-Eclat) [90] were developed, respectively. Research efforts have already been made to improve Apriori-based and traditional FIM algorithms and to convert them into parallel versions, mostly under the MapReduce [18] or Spark [61] environment. Some results of these efforts are PARMA [20], a parallel frequent itemset mining algorithm with Spark (R-Apriori) [91], PaMPa-HD [27], and PHIKS [28]. PARMA [20] is a parallel algorithm for the MapReduce framework, which mines approximate association rules instead of exact ones.

Discussions. The above parallel algorithms are used in frequent itemset mining. Up to now, large numbers of studies of parallel frequent itemset mining have been extensively studied compared to parallel sequential pattern mining. The development of PFIM is still an active area in research, where the up-to-date advances of PFIM can be referred to [17], [38]. As mentioned before, efficient mining of frequent sequences is more challenging than FIM. Currently, a large number of algorithms have been extensively developed for SPM, while few of them are sufficiently scalable to handle large-scale datasets.

4.2 Partition-based Algorithms for PSPM

In addition to parallel frequent itemset mining, a variety of mining algorithms has been developed for parallel sequential pattern mining (PSPM). Although PSPM is comparatively more complicate and challenging than that of PFIM, research in the area of PSPM has been active for some time. Recently, to improve the performance, effectiveness and scalability issues, parallel sequential pattern mining has received much attention. As shown in Table 8, some partition-based parallel methods for SPM have been developed.

A comprehensive working example of partition-based sequential pattern mining algorithm with task parallelism is illustrated below, to show how the problem of parallel sequential pattern mining is solved. It could help to understand the idea of parallel sequential pattern mining.

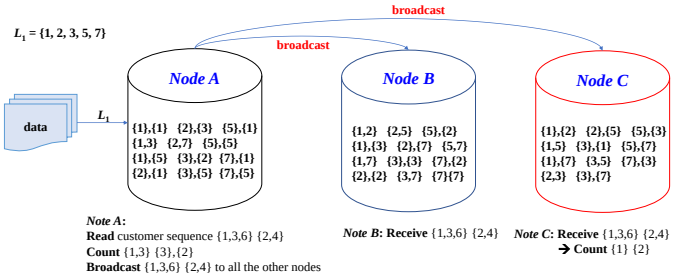

Fig. 2. An illustrated example of the SPSPM algorithm [92].

Note that Fig. 2 is an illustrated example of the SPSPM algorithm (Simply Partitioned Sequential Pattern Mining with task parallelism) [92]. Assuming there are 3 nodes in the mining system, denoted as *Node A*, *Node B* and *Node C*, respectively. Each node reads the sequence database D_k from its local disk, and let the large/frequent items be $L_1 = \{1, 2, 3, 5, 7\}$. Since SPSPM partitions/distributes the candidate sequences in a round-robin manner, the candidate sequences are assigned shown in Fig. 2. Suppose *Node A* reads the sequence $s = \{1, 3, 6\}\{2, 4\}$. Then *Node A* broadcasts s stored in its local disk to all the other nodes, and increases the count-support of $\{1, 3\}, \{3\}\{2\}$. *Node B* and *Node C* respectively receive the sequence $\{1, 3, 6\}, \{2, 4\}$ from *Node A*. Then *Node C* increases the support-count of $\{1, 2\}$.

NPSPM, SPSPM and HPSPM. In 1998, Shintani and Kitsuregawa firstly partitioned the input sequences for SPM, and yet they assumed that the entire candidate sets could be replicated and would fit in the overall memory (random access memory and hard drive) of a process [92]. Based on the classical GSP approach [5], they proposed three parallel approaches [92]: NPSPM (Non-Partitioned Sequential Pattern Mining), SPSPM (Simply Partitioned Sequential Pattern Mining), and HPSPM (Hash Partitioned Sequential Pattern

Mining). These three variants perform as a classical level-wise candidate generation-and-test style. HPSPM is similar to SPSPM but utilizes more intelligent strategies. HPSPM uses the hash mechanism to assign the input sequences and candidate sequences to specific processes [92]. Although HPSPM has the best performance among the three variants, it also suffers, however, several limitations. For exampe, when the object time-lines are very large, or when the counting method for generalized SPM is used, HPSPM is very time-consuming and memory cost.

EVE & EVECAN. Similar assumptions of NPSPM were made in EVE [93] and EVECAN [93]. EVECAN (EVEnt and CANdidate distribution) partitions the input sequentail data similar to EVE (EVEnt distribution), and also partitions the candidate sequences. In EVECAN, both the candidate generation phase and the counting phase are parallelized. The former uses a distributed hash table mechanism, whereas the later adopts an approach similar to Intelligent Data Distribution (IDD), which is usually used for non-sequential associations. EVECAN shards both input sequences and candidate sequences, and rotates the smaller of the two sequence sets among the processes, in round-robin fashion. Note that the above parallel formulations for SPM only toward the shared-memory architecture.

DPF, STPF and DTPF. Guralnik and Karypis developed three tree-projection-based parallel sequence mining algorithm [94], named data parallel formulation (DPF), static task-parallel formulation (STPF), and dynamic task-parallel formulation (DTPF), for distributed-memory architectures. These projection algorithms are parallelized using the data parallel formulation (DPF) and the task parallel formulation (TPF). They are intrinsically similar to the PrefixSpan algorithm [64], grow the lexicographic sequence tree in a bi-level breadth-first manner. Two partitioning methods were proposed: TPF-BP (TPF based on a bin-packing algorithm) and TPF-GP (TPF based on a minimum-cut bipartite graph partitioning algorithm), both TPF-BP and TPF-GP can achieve balanced workloads. It was shown that these three algorithms are able to achieve good speedups when the number of processors increases. However, STPF may suffer some load imbalance problem while the number of processes increases, and DPF has not only to partition the tree, but also needs to perform multiple scans of the local database at each iteration. The added I/O overhead makes DPF impractical for big data analytics. In addition, DPF utilizes a dynamic load-balancing strategy to allow an idle processor to join the busy processors. However, this dynamic strategy requires much more inter-processor communication than the selective sampling approach used in the Par-CSP algorithm [34]. Moreover, the interruption of these busy processors may even cause more overhead during the mining process.

Summary of merits and limitation. In this subsection, we have described the key details of some partition-based algorithms for PSPM so far. We further highlight the key differences, advantages and disadvantages between these approaches, as shown in Table 8.

4.3 Apriori-based Algorithms for PSPM

As the above partition-based algorithms of PSPM, input partitioning is not inherently necessary for shared memory systems or MapReduce distributed systems. In this subsection, some typical Apriori-based algorithms for PSPM are introduced. Moreover, the key differences, advantages and disadvantages between these approaches are highlighted at Table 9.

pSPADE and webSPADE. Zaki first extended the efficient SPADE algorithm to the shared memory parallel architecture, called pSPADE [31]. In the pSPADE framework, input sequence data is assumed to be resided on shared hard drive space, and stored in *lattice* (a vertical data structure). pSPADE utilizes an optimized task parallel formulation approach (Recursive Dynamic Load Balancing, RDLB), and uses two data parallel formulations (single vs. level-wise id-list parallelism and join parallelism), to partition the input spaces. Processes are either assigned id-lists for a subset of sequences, or portions of all id-lists associated with a range of sequences in database. After that, processes collaborate to expand each node in the itemset *lattice*. However, these formulations lead to poor performance due to high synchronization and memory overheads. Zaki then proposed two task distribution schemes, which are able to avoid read/write conflicts through independent search space sub-lattice assignments. Besides, a web-based parallel approach called webSPADE is proposed to analyze the web log [95]. webSPADE is also a SPADE-based algorithm to be run on shared memory parallel computers.

DGSP and DPSP. A distributed GSP algorithm based on MapReduce [18] framework named DGSP was introduced in [96]. The "two-jobs" structure used in DGSP can effectively reduce the communication overhead for parallelly mining sequential patterns. DGSP optimizes the workload balance, partitions the database, and assigns the fragments to Map workers. All in all, DGSP is a straight-forward extension of GSP in distributed environment, it performs poorly because of the repeated scans of the input data. Huang et al. developed Map/Reduce jobs in DPSP to delete obsolete itemsets, to update current candidate sequential patterns, and to report up-to-date frequent sequential patterns within each period of interest (POI) [97]. One major drawback of DPSP is that it needs to proceed through numerous rounds of MapReduce's jobs, which will easily cause the load unbalancing problem and incur high cost of MapReduce initialization. Anthor drawback is that it focuses on the POI concept, where the database is divided into many POI windows. Thus, its ability of handling longer sequential patterns is not demonstrated.

PartSpan and GridGSP. After that, some distributed and parallel mining methods, such as parallel sequence mining of trajectory pattern (PartSpan) [98] and GridGSP [99], were proposed to find trajectory patterns by extending the traditional GSP algorithm [5]. The PartSpan, which is based on GSP and HPSPM, adopts the prefix-projection and the parallel formulation [98]. Thus, it can efficiently solve the I/O cost by using the candidate pruning strategy and reasonable data distribution [98]. GridGSP is a distributed and parallel GSP algorithm in a grid computing environment, it provides a flexible and efficient platform for SPM [99]. In GridGSP, the input sequence data is divided into a number of progressive windows, and then these data independently perform candidate generation on multiple machines. Then, reducer uses the support assembling jobs

TABLE 8
Partition-based algorithms for parallel sequential pattern mining.

Name	Description	Pros.	Cons.	Year
HPSPM [92]	Hash Partitioned Sequential Pattern Mining, task parallelism.	• It can considerably reduces the broadcasting of sequences.	• It is based on GSP and has a level-wise candidate generation-and-test style.	1998
SPSPM [92]	Simply Partitioned Sequential Pattern Mining, task parallelism.	• The candidate sets assigned to the nodes have almost the same size. • They are disjoint from each other.	• It requires a very large communication overhead.	1998
NPSPM [92]	Non-Partitioned Sequential Pattern Mining, data parallelism.	• Mining sequential patterns without partition.	• NPSPM is based on GSP and performed in a bottom-up, level-wise manner with multiple data scans.	1998
EVE [93]	EVEnt distribution.	• Assume that the entire candidate sequences can be replicated and fit in the overall memory of a process.	• The I/O overhead is impractical in the big data context.	1999
EVECAN [93]	EVEnt and CANdidate distribution.	• Rotates the smaller of two sets among the processes with a round-robin fashion.	• The I/O overhead is impractical in the Big Data context.	1999
DPF, STPF and DTPF [94]	Data Parallel Formulation, Static Task-Parallel Formulation, and Dynamic Task-Parallel Formulation	• It can achieve balanced workloads using a bi-level breadth-first manner in the lexicographic sequence tree.	• At each iteration, it has to partition the tree and perform multiple scans of the local database. • This added I/O overhead is impractical when mining big data.	2004

to count the relative occurrence frequencies of candidate sequences. Besides, GridGSP uses the Count Distribution (CD) technology for counting 1-length candidate sequences, followed by a projection-based task partitioning for solving the remainder of the PSPM problem. The similar strategies were used in PLUTE [100] and MG-FSM [101]. Both PartSpan and GridGSP focus on efficiently mining the frequent patterns under different domain-specific constraints or on different types of data by using MapReduce.

SPAMC & SPAMC-UDLT. Recently, Huang et al. extend the classic SPAM algorithm to a MapReduce version, named as SPAM algorithm on the Cloud (SPAMC for short). Cloud-based SPAMC utilizes an iterative MapReduce to quickly generate and prune candidate sequences while constructing the lexical sequence tree [102]. The core technology of SPAMC is that it mines a sequential database using parallel processing, and the all sub-tasks can be simultaneously distributed and executed on many machines. However, SPAMC does not address the iterative mining problem, as well as the load balancing problem. Thus, reloading data in SPAMC incurs additional costs. Although SPAMC utilizes a global bitmap, it is not scalable enough for handing extremely large-scale databases. To remedy these problems, Chen et al. further devised SPAMC-UDLT to discover the sequential patterns in the cloud-uniform distributed lexical sequence tree, exploiting MapReduce and streaming processes [103]. SPAMC-UDLT dramatically improves overall performance without launching multiple MapReduce rounds, and provides perfect load balancing across machines in the cloud [103]. The results [103] showed that SPAMC-UDLT can remarkably reduce execution time compared to SPAMC, achieve high scalability, and provide much better load balancing than existing algorithms in the cloud.

Summary of merits and limitation. All the above-described algorithms, which are shown in Table 9, are the Apriori-based algorithms for parallel sequential pattern mining. In general, all these algorithms may also suffer from some drawbacks of serial Apriori-based SPM algorithm. Thus, they do not scale well and can not efficiently solve the problem such as time-consuming and memory cost.

4.4 Pattern Growth Algorithms for PSPM

As mentioned before, many pattern growth algorithms for serial sequential pattern mining have been extensively studied. Based on these theories and techniques, some of these algorithms have been extended to realize the parallelization. As shown in Table 10. It contains key characteristics, advantages and disadvantages of the pattern growth approaches. The PrefixSpan algorithm also has some parallelized versions, such as Par-ASP [104] and Sequence-Growth [107].

Par-FP, Par-ASP and Par-CSP. In order to balance mining tasks, Cong et al. designed several pattern-growth models, Par-FP (Parallel FP-growth) [104], Par-ASP (Parallel PrefixSpan to mine all sequential-patterns) [104] and Par-CSP (Parallel mining of Closed Sequential Patterns) [34], to accomplish the static task. Especially, they use a sampling technique, named *selective sampling*, that requires the entire input data be available at each process. Par-ASP is extended by the classical PrefixSpan algorithm [64], and the used *selective sampling* in Par-ASP is performed by four steps: sample tree construction, sample tree mining, task partition and task scheduling. After collecting 1-length frequent sequence counts, Par-ASP separates a sample of k-length frequent prefixes of sequences in database. It then uses one process to handle the sample via a pattern growth algorithm, recording the execution times for the found frequent subsequences. Correspondingly, mining each of frequent subsequences can be transformed into a parallel task. A series of projected sub-databases for these frequent sub-sequences

TABLE 9
Apriori-based algorithms for parallel sequential pattern mining.

Name	Description	Pros.	Cons.	Year
pSPADE [31]	Parallel SPADE.	• RDLB exploits the dynamic load balancing at both inter-class and intra-class granularities.	• It mines patterns with level-wise candidate generation-and-test.	2001
webSPADE [95]	A parallel-algorithm variation of SPADE developed for web logs.	• It requires once scan of the input data and multi-processor servers. • Data and task parallelism are achieved easily by multi-threaded programming.	• As a Apriori-based, vertical formatting method similar to SPADE, it needs several partial scans of the database.	2002
DGSP [96]	A distributed GSP algorithm based on MapReduce.	• DGSP optimizes the workload balance, partitions the database, and assigns the fragments to Map workers.	• It may performe poorly because of the repeated scans of the input sequence data.	2005
PartSpan [98]	Parallel sequence mining of trajectory patterns.	• It can reduce the I/O cost by using the candidate pruning strategy and reasonable data distribution.	• It does not scale well and can not efficiently solve the problem of time-consuming and memory cost.	2008
DPSP [97]	A sequence algorihtm that mining Distributed Progressive Sequential Patterns on the cloud.	• It performs a progressive mining process in a dynamic database. • It can discover up-to-date frequent sequential patterns.	• It may cause the load unbalancing problem because of the required numerous rounds of MapReduce jobs. • It may easily incur high cost of MapReduce initialization.	2010
GridGSP [99]	A distributed and parallel GSP in a grid computing environment.	• It utilizes the divide-and-conquer strategy. • Several monitoring mechanisms are developed to help manage the SPM process.	• Suffer some drawbacks of the GSP algorithm, such as time-consuming and memory cost.	2012
SPAMC [102]	SPAM algorithm on the Cloud.	• Could-based SPAMC utilizes an iterative MapReduce framework to increase scalability.	• It is not an iterative mining approach, and may incur additional costs for reloading data. • It does not study the load balancing problem. • It is still not scalable enough.	2013
SPAMC-UDLT [103]	SPAM algorithm in the cloud-uniform distributed lexical sequence tree.	• It is an iterative mining approach. • It studies the load balancing problem. • Exploiting MapReduce and streaming processes.	• It does not consider the gap constraint. • It does not study condense representation (i.e., closed, maximal).	2013

are done while estimating their execution times. After that, task distribution is done in the same way as in the Par-FP algorithm [104]. However, it has been found that the serial sampling operations of these algorithms limit the speedup potential when the number of processes increases.

As described at above subsection, some closed-based SPM algorithms have been extensively studied. Unfortunately, no parallel method for closed sequential pattern mining has yet been proposed. In 2005, Cong et al. first proposed the Par-CSP (Parallel Closed Sequential Pattern mining) algorithm [34] based on the BIDE algorithm. Par-CSP [34] is the first parallel algorithm which aims at mining closed sequential patterns. The process in Par-CSP is divided into independent tasks, this can minimize inter-processor communications. Par-CSP utilizes a dynamic scheduling strategy to reduce processor idle time and is performed in a distributed memory system. It is different from pSPADE using a shared-memory system. Par-CSP also uses a load-balancing scheme to speed up the process. Applying such a dynamic load balancing scheme in a distributed memory system is too expensive to be practical even for big data. In contrast to mine closed sequences, a parallel version of the MSPX [81] algorithm named PMSPX was also proposed. It mines maximal frequent sequential patterns by using multiple samples [108].

In addition, sequential mining can be applied into biological sequence mining, such as a protein DNA analysis. Recently, the 2PDF-Index [109], 2PDF-Compression [109], and DFSP [106] algorithms were proposed and applied to scalable mining sequential patterns for biological sequences. DFSP [106] uses a three-dimensional list for mining DNA sequences. It adopts direct access strategy and binary search to index the list structure for enumerating candidate patterns. However, these three works focus on efficiently mining the frequent patterns under different domain-specific constraints or on different types of data (e.g., medical database, DNA sequences), while SPAMC-UDLT [103] focuses on the general cases without any constraints.

Summary of merits and limitation. According the above analytics, it can be concluded that the pattern-growth approaches usually perform better than those of Apriori-based algorithms for PSPM. Moreover, the key differences, advantages and disadvantages between these approaches are highlighted in Table 9.

TABLE 10
Pattern growth algorithms for parallel sequential pattern mining.

Name	Description	Pros.	Cons.	Year
Par-ASP [104]	Parallel PrefixSpan to mine all sequential patterns (ASP).	• It uses a sampling method to accomplish static task partition.	• The serial sampling component limits the speedup potential when the number of processes increases.	2005
Par-CSP [34]	Parallel mining Closed Sequential Patterns with selective sampling.	• The first parallel algorithm for mining closed sequential patterns. • It can reduce processor idle time using a dynamic scheduling scheme. • Using a load-balancing scheme.	• The serial sampling component limits the speedup potential when the number of processes increases.	2005
PLUTE [100]	Parallel sequential pattern mining.	• It focuses on massive trajectory data, and outperforms PartSpan in mining massive trajectory data.	• The used prefix projection technology could consume huge memory and a lot of scan time.	2010
BIDE-MR [105]	A BIDE-based parallel algorithm on MapReduce.	• The tasks of closure checking and pruning can be iteratively assigned to different nodes in cluster. • Performed on Hadoop cluster with high speed-ups. • Can mine closed sequential patterns.	• Even mining frequent closed sequences does not fully resolve the problem of pattern explosion.	2012
DFSP [106]	A Depth-First SPelling algorithm for mining sequences from biological data.	• It proposes a three-dimensional list for mining DNA sequences. • It addresses biological data mining, such as protein DNA analysis.	• It focuses on PSPM under different domain-specific constraints, but not the general cases without any constraints.	2013
Sequence -Growth [107]	A MapReduce-based FIM algorithm with sequences.	• It uses a lexicographical order to generate candidate sequences that avoiding expensive scanning processes. • It provides an extension for trajectory pattern mining.	• It does not consider the gap-constraint and support hierarchies.	2015

4.5 Hybrid Algorithms for PSPM

Some hybrid parallel algorithms for SPM are developed by incorporating the above different technologies, as shown in Table 11. Details of some important hybrid algorithms are described below.

MG-FSM and MG-FSM+. MG-FSM [101] and its enhanced version MG-FSM+ [58] are the scalable distributed (i.e., shared-nothing) algorithms for gap-constrained SPM on MapReduce [18]. In MG-FSM and MG-FSM+, the notion of w-equivalency w.r.t. "projected database" which is used by many SPM algorithms is introduced. Moreover, some optimization techniques are developed to minimize partition size, computational costs, and communication costs. Therefore, MG-FSM and MG-FSM+ are more efficient and scalable than previous alternative approaches. Especially, any existing serial FSM method can be applied into MG-FSM to handle each of its partitions. MG-FSM is divided into three key phases: i) a preprocessing phase, ii) a partitioning phase, and iii) a mining phase; all these phases are fully parallelized. It is more adaptable and flexible that both MG-FSM [101] and MG-FSM+ [58] can handle temporal gaps and maximal and closed sequences, which improves their usability.

MG-FSM vs. LASH. In some real-world applications, hierarchy and frequency of items can be different, while MG-FSM does not support hierarchies. To overcome this drawback, LASH is recently designed for large-scale sequence datasets. It is the first parallel algorithm to discovery frequent sequences by considering hierarchies [30]. Inspired by MG-FSM, LASH first partitions the input data, and then handles each partition independently and in parallel. Both of them are the distributed algorithms for mining SPs with Maximum gap and maximum length constraints using MapReduce [18], they are all more adaptable and flexible than other PSPM algorithms. Notice that LASH does not have a global post-processing, it divides each sequence by pivot item and performs local mining (PSM). Therefore, LASH can have better search performance than MG-FSM and MG-FSM+.

ACME. For the application in bioinformatics, Sahli et al. proposed ACME [113], which applies tree structure to extract longer motif sequences on supercomputers. ACME is a parallel combinatorial method for extracting motifs repeated in a single long sequence (e.g., DNA) instead of multiple long sequences. It proposes two novel models, CAST (cache aware search space traversal model) and FAST (fine-grained adaptive sub-tasks), to effectively utilize memory caches and processing power of multi-core shared-memory machines. ACME introduces an automatic tuning mechanism that suggests the appropriate number of CPUs to utilize. In particular, it is large-scale shared nothing systems with tens of thousands of processors, which are typical

TABLE 11
Hybrid algorithms for parallel sequential pattern mining.

Name	Description	Pros.	Cons.	Year
MG-FSM [101]	A large-scale algorithm for Frequent Sequence Mining on MapReduce with gap constraints.	• Uses some optimization techniques to minimize partition size, computational cost, and communication cost. • More efficient and scalable than other alternative approaches. • Any existing SPM method can be used to mine one of its partitions. • Can handle temporal gaps, maximal and closed sequences, which improves its usability.	• MG-FSM does not support hierarchies, that some of items have hierarchies and frequency can be different.	2013
DFSP [106]	A Depth-First SPelling algorithm for mining sequences from biological data.	• It proposes a three-dimensional list for mining DNA sequences. • It addresses biological data mining, such as protein DNA analysis.	• It focuses on PSPM under different domain-specific constraints, but not the general cases without any constraints.	2013
IMRSPM [110]	Uncertain SPM algorithm in iterative MapReduce framework.	• An iterative MapReduce framework for mining uncertain sequential patterns. • The first work to solve the large-scale uncertain SPM problem.	• It extends the Apriori-like SPM framework to MapReduce, thus suffers from some drawbacks of Apriori-like SPM algorithms.	2015
MG-FSM+ [58]	An enhanced version of MG-FSM.	• A more scalable distributed (i.e., shared-nothing) sequential pattern mining algorithm. • A more suitable SPM approach to directly integrate the maximality constraint.	• Similar to MG-FSM, MG-FSM+ does not support hierarchies either.	2015
LASH [30]	A LArge-scale Sequence mining algorithm with Hierarchies.	• It is the first parallel algorithm for mining frequent sequences with consideration of hierarchies. • It proposes the pivot sequence miner (PSM) method for mining each partition. • LASH can search better than MG-FSM because of PSM. • LASH does not need a global post-processing.	• It cannot handle stream data and uncertain data.	2015
Distributed SPM [111]	A distributed SPM approach with dynamic programming.	• A memory-efficient approach uses distribute dynamic programming schema. • Uses an extended prefix-tree to save intermediate results. • Its time cost is linear.	• It does not consider the gap-constraint and support hierarchies.	2016
Interesting Sequence Miner (ISM) [112]	A novel subsequence interleaving parallel model based on a probabilistic model of the sequence database.	• It takes a probabilistic machine learning approach (an encoding scheme) to the SPM problem. • All operations on the sequence database are trivially parallelizable.	• It cannot handle stream data and uncertain data.	2015

in cloud computing. It has been shown that ACME achieves improvement in serial execution time by almost an order of magnitude, especially supports for longer sequences (e.g., DNA for the entire human genome).

Summary of merits and limitation. In most PSPM algorithms, they are hybrid inherently. Although the hybrid methods for PSPM incorporate different technologies to effectively reduce communication cost, memory usage and execution time. There is no doubt that they also have their advantages and disadvantages. Details of characteristics, advantages and disadvantages about these hybrid methods for PSPM are described at Table 11.

5 Advanced Topics in PSPM

In this section, some advanced topics of parallel sequential pattern mining are provided and discussed in details, including parallel quantitative / weighted / utility sequential pattern mining, parallel sequential pattern mining from uncertain data and stream data, hardware accelerator for PSPM.

5.1 Parallel Quantitative / Weighted / Utility Sequential Pattern Mining

Generally speaking, most PSPM algorithms focus on how to improve their efficiency. In many cases, effectiveness can be

far more important than efficiency. It is commonly seen that sequence data contains various quantity, weight, and utility with different items/objects [82], [114]. For example, consider some transactions from shopping baskets. As shown in Table 3, the quantity and price of each item are not considered in this database. In general, some useful information (i.e., quantitative, weight and utility) are simply ignored in the current form of sequential pattern mining. Up to now, some approaches have been proposed to handle quantity, weight or utility constraints in sequential pattern mining. Many sequential pattern algorithms have been introduced, such as quantitative sequential pattern [82], [115], weighted sequential pattern [116], high utility sequential pattern [114], etc. All of these efforts rely on sequential algorithms and are not able to scale to larger-scale datasets. How to achieve parallelism of these methods is an open issue and challenge.

Recently, the methods for mining above useful sequential patterns are extended to achieve parallelism, such as the PESMiner (Parallel Event Space Miner) framework [117] for parallel and quantitative mining of sequential patterns at scale, and the Spark-based BigHUSP algorithm [118] for mining high utility sequential patterns. BigHUSP is first proposed by Zihayat et al., it is designed based on the Apache Spark platform [61] and takes advantage of several merit properties of Spark such as distributed in memory processing, fault recovery and high scalability [118]. On one hand, both PESMiner [117] and BigHUSP [118] can identify informative sequential patterns with respect to a business objective, such as interval-based quantitative patterns and profitable patterns. Thus, they may be more effective than those which simply ignore the useful information. On the other hand, these frameworks integrate domain knowledge, computational power, and MapReduce [18] or the Apache Spark [61] techniques together to achieve better feasibility, quality and scalability.

5.2 PSPM in Uncertain Data

In many real-life applications, uncertain data is commonly seen, and uncertain sequence data is widely used to model inaccurate or imprecise timestamped data [119], while traditional datat mining (i.e., FIM, ARM, SPM) algorithms are inapplicable when handling uncertain data. Other related algorithms for parallel sequential pattern mining are still being developed, such as the PSPM of large-scale databases by using a probabilistic model [112]. Recently, Ge at al. proposed an iterative MapReduce framework for mining uncertain sequential patterns, where the new iterative MapReduce-based Apriori-like uncertain SPM framework is quite different from the traditional SPM algorithms [110]. An uncertain sequence database $D = \{d_1, ..., d_n\}$ is abstracted by a Resilient Distributed Datasets (RDD) [61] in Spark. Uncertain sequences in the RDD are allocated to a cluster of machines and can be processed in parallel. A sequential pattern s in D is called a probabilistic frequent pattern (*p-FSP*) if and only if its probability of being frequent is at least ψ_p, denoted by $P(sup(s) \geq \psi_s) \geq \psi_p$. Here, ψ_s is the user-defined minimum probability threshold. However, the frequentness of s in an uncertain database is probabilistic.

For application of SPM in uncertain databases [119], it is challenging in terms of efficiency and scalability. Ge and Xia further developed a Distributed Sequential Pattern (DSP) mining algorithm in large-scale uncertain databases based on Spark, which relies on a memory-efficient distributed dynamic programming (DP) approach [111]. Although MapReduce [18] is widely used for processing big data in parallel, while it does not support the iterative computing model, its basic framework cannot be directly used in SPM. Directly applying the DP method to Spark is impractical because its memory-consuming characteristic may cause heavy JVM garbage collection overhead in Spark [111]. For bioinformatics, a new spark-based framework was proposed to mine sequential patterns from uncertain big deoxyribonucleic acid (DNA) [120].

5.3 PSPM in Stream Data

Most of the above PSPM algorithms are developed to handle traditional database management system (DBMS)[8] where data are stored in finite, persistent databases. For some real-life applications, it is commonly seen to process the continuous data stream which is quite different from DBMS. Stream data is temporally ordered, fast changing, massive, and potentially infinite [121]. Few stream algorithms have been developed for parallel mining sequential patterns up till now. Recently, Chen et al. proposed two parallel sequential pattern mining algorithms, SPAMC [102] and SPAMC-UDLT [103], for processing stream data. Since SPAMC is still not scalable enough for mining large-scale data, the SPAMC-UDLT algorithm [103] was further proposed to address the fast changing massive stream data. Notice that the characteristics of SPAMC and SPAMC-UDLT have been highlighted at Table 9, and some comments and differences about them have already been mentioned before. The advantages of these cloud-based algorithms are that they can scan the stream data only once, and can effectively generate all the frequent sequential patterns by using stream mining operation. In cases of multiple streams in parallel, the MSSBE algorithm [120] can find sequential patterns in a multiple-stream environment, where pattern elements can be part of different streams.

5.4 Hardware Accelerator for PSPM

Generally, many efforts have been made to speed up SPM via software and hardware. On the one hand, parallel implementation is the general way. On the other hand, hardware accelerators allow a single node to achieve orders of magnitude improvements in performance and energy efficiency. As mentioned before, Graphics processing units (GPUs) [21], [39] have attracted much attention due to their cost-effectiveness and enormous power for massive data parallel computing. General-purpose GPUs leverage high parallelism, but GPUs' single instruction multiple data (SIMD) and lockstep organization means that the parallel tasks must generally be similar. Lin et al. developed a novel parallel algorithm on GPUs for accelerating pattern matching [60]. Based on the traditional AC algorithm, they implement three CPU versions and one GPU version as follows: ACCPU, DPACOMP, PFACOMP, and PFACGPU [60].

8. https://en.wikipedia.org/wiki/Database

Then Memeti and Pllana introduced an approach for accelerating large-scale DNA sequence analysis and SIMD parallelism using many-core architectures (such as GPU, Intel Xeon Phi coprocessor) [122]. Recently, Wang et al. proposed a hardware-accelerated solution, AP-SPM and GPU-GSP, for sequential pattern mining (SPM) [123]. The major ideas are summarized as follows: It uses Micron's new Automata Processor (AP), which provides native hardware implementation of non-deterministic finite automata. Based on the well-known GSP approach, AP SPM derives a compact automaton design for matching and counting frequent sequences. According to experiments with a single-threaded CPU, multi-core CPU, and GPU GSP implementations [123], it has been shown that the AP-accelerated solution outperforms PrefixSpan [64] and SPADE [68] on multi-core CPU, and scales well for larger datasets.

6 Open-Source Software

Although the problem of sequential pattern mining has been studied for more than two decades, and the advanced topic of parallel sequential pattern mining also has been extended to many research fields, few implementations or source code of these algorithms are released. This brings some barriers to other researchers that they need to re-implement algorithms for using the algorithms or comparing the performance with novel proposed algorithms. To make matter worse, it may cause the unfairness while running experimental comparison, since the performance of pattern mining algorithms (i.e., FIM, ARM and SPM) may commonly depending on the used compiler and the used machine architecture. We provide some open-source software of SPM and PSPM in Table 12.

- **Sequence mining**. Zaki releases many implementations or source code of data mining algorithms on his Website, including itemset mining and association rules, sequence mining, itemset and sequence utilities, tree mining, graph mining & indexing, and clustering. He provides C code for three sequence mining algorithms.
- **SPAM**. There is a special website (as shown in Table 12), named "Himalaya Data Mining Tools", built for the classical SPAM algorithm [23] which aims at finding all frequent sequences. It provides an overview, an illustrative example, source code and documentation, as well as the presentation slides of SPAM.
- **SPMF**. SPMF [80] is a well-known open-source data-mining library, which implements many algorithms and has been cited in more than 600 research papers since 2010. SPMF is written in Java, and provides implementations of 120 data mining algorithms, specialized in sequential pattern mining. SPMF has the largest collection of implementations of various algorithms for sequential pattern mining. It provides the Java code for AprioriAll [2], GSP [5], PrefixSpan [64], SPADE [68], SPAM [23] and many others. The only problem is that SPMF does not provide any parallel algorithms.
- **MG-FSM & MG-FSM+**. MG-FSM [101] is a scalable, general-purpose (e.g., general, with Maximum gap constraints, and with maximum length constraints, etc.) frequent sequence mining algorithm built for MapReduce. The Java implementation and command line options of MG-FSM are provided at Github.
- **LASH**. LASH [30] is a new high scalable, MapReduce-based distributed sequence mining algorithm. It mines sequential patterns by considering items' hierarchies. The Java source code of LASH is provided at Github, including prerequisites for building LASH and running instructions.

7 Challenges and Opportunities in Parallel Sequential Pattern Mining

7.1 Challenges in PSPM

In recent decades, many models and algorithms have been developed in data mining to efficiently discover the desired knowledge from various types of databases. There are still, however, many challenges in big data mining. According to Labrinidis and Jagadish's study, the major challenges in big data analysis include: data inconsistency, data incompleteness, scalability, timeliness, and security [42]. Although useful developments have been made in the field of parallel computing platforms, there are still some technical challenges [24], [44]. Despite advances in the field of parallel data mining, especially parallel frequent itemset mining and parallel sequential pattern mining, both in academia and industry, significant challenges still remain in parallel sequential pattern mining. Some challenges which need to be dealt with for realizing parallel sequential pattern mining are respectively explored in the following section.

(1) ***Complex types of sequence data.*** Sequential data commonly occurs in many fields, current developments in parallel sequential pattern mining have successfully addressed some types of sequence data, including the general type of sequence, sequence data containing quantity / weight / utility, uncertain sequential data, stream data, and so on. The related algorithms and some extensions have been mentioned previously. However, there are still various more complex types of sequence data in a wide range of applications. For future research, it is a big challenge to develop more adaptable and flexible PSPM frameworks for dealing with more complex types of sequence data in a parallel environment.

(2) ***Multi-modal data.*** In the big data era, a key property of multimodality is complementarity. Each modality brings to the whole some type of added value that cannot be deduced or obtained from any of the other modalities in the setup. In mathematical field, this added value is so called diversity. Thus, multimodality and data fusion are a potential challenge to deal with big data before using parallel sequential pattern mining algorithms.

(3) ***Dynamic sequence data.*** In a wide range of applications, the processed data may be commonly dynamic but not static [121]. The dynamic data is more complicated and difficult than the static data. Although some approaches have been developed for handling dynamic sequence data, such as incremental mining [75], [124], [125], online progressive mining [126], and stream data processing [103], [121]. There are still many challenges when parallelizing these mining methods, especially for parallelizing dynamic sequential pattern mining algorithms.

(4) ***Scalability.*** In general, parallelizing SPM algorithms for dealing with big data is more complicated and difficult

TABLE 12
Open-source software for SPM or PSPM.

Name	Contributors	Website
Sequence Mining	Zaki et al.	http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software
SPAM [23]	Gehrke et al.	http://himalaya-tools.sourceforge.net/Spam
SPMF [5]	Fournier-Viger et al.	http://www.philippe-fournier-viger.com/spmf
MG-FSM & MG-FSM+ [69]	Miliaraki et al.	https://github.com/uma-pi1/mgfsm
LASH [30]	Beedkar et al.	http://uma-pi1.github.io/lash

than that of small data. It easily brings some challenging problems, such as availability, accuracy, and scalability. When dealing with big data, the disadvantages of current data mining techniques are centered on inadequate scalability and parallelism. Scalability is one of the core technologies to meet these challenges. Thus, there is a major challenge to realize the needed scalability of the developed PSPM algorithm with demonstrated elasticity and parallelism capacities.

(5) ***Privacy.*** Data has the inestimable value (i.e., hidden knowledge and more valuable insights). Data mining and analytics offer many useful tools and play an important role in many fields. However, a major risk in data management and data mining is data leakage, which seriously threatens privacy. Public concerns regarding privacy are rising, which is a major concern in data mining, especially big data mining. On one hand, policies that cover all user privacy concerns should be developed. On the other hand, privacy preserving data mining (PPDM) [127] is also needed. Up to now, many technologies and algorithms have been developed for PPDM [127], but few of them are related to parallel sequential pattern mining. With the developments of SPM to analyze personal data, it is quite necessary and challenging to address the privacy preserving problem for parallel sequential pattern mining.

7.2 Opportunities in PSPM

There are a variety of traditional, distributed, and parallel data mining algorithms in numerous real-life applications. To meet the demand of large-scale and high performance computing, the problem of parallel/distributed data mining has received considerable attention over the past decade. There are many types of parallel and distributed systems, such as multi-core computing [55], grids [47], [48], peer-to-peer (P2P) systems [49], ad-hoc networks [50], cloud computing systems [51], and the MapReduce framework [18]. All these developments can provide theoretical and technical support for parallel sequential pattern mining. Some opportunities are summarized below.

(1) ***New applications***. According to the research of current status, most algorithms of PSPM focus on how to improve the mining efficiency. It is commonly seen that sequential data occurs in many fields such as basket analysis, sensor networks, bioinformatics, social network analysis, and the internet of things. There are some important research opportunities for PSPM in these fields. Instead of focusing on faster general PSPM algorithms, the trend clearly goes towards to extend parallel sequential pattern mining in new applications, or in new ways for existing applications [8].

(2) ***Advanced parallel computing environment***. To overcome the scalability challenge of big data, several attempts have been made in exploiting massive parallel processing architectures, e.g., cloud [51], MapReduce [18], Spark [61]. All these computing infrastructures provide new opportunities for the design of parallel data mining, especially for parallel sequential pattern mining algorithms. To some degree, all these developments can provide the necessary theoretical and technical support for parallel sequential pattern mining.

(3) ***Developments from hardware and software***. As mentioned before, a trend of parallel computing is graphics processing unit (GPU) based parallel techniques. Architecturally, a GPU is composed of hundreds of cores that can simultaneously handle thousands of threads/tasks [21], [39]. More and more supercomputers have the GPU component which is different from the traditional CPU. Therefore, we can expect an enhancement on parallel data mining with the progress of powerful computational capabilities. For future research, the developments from hardware and software strongly provide some opportunities for the sequence mining task in a parallel environment.

(4) ***Keeping pattern short and simple***. Although there have been numerous studies on pattern mining (e.g., FIM, ARM, SPM, etc.), the study on interesting pattern mining is still limited. Most of them are only limited to discover the complete set of desired patterns, a huge pattern mining results may still confuse users to make inefficient decisions. What iss worse, they may easily cause redundant results when mining long patterns or with a low support threshold. Returning a condensed or more constrained set is an interesting issue. Some methods of condensed SPM (i.e., CloSpan [66], BIDE [65], MSPX [81] and ClaSP [83], as shown at Table 7) or summarization SPM (i.e., summarizing event sequences [128], SQUISH [129]) have been proposed, that is, instead of returning all patterns that occur more often than a certain threshold, these methods only return a condensed or more constrained set. How to achieve parallelism of these condensed or summarizing methods is an open issue and challenge.

(5) ***Utilizing deep learning technology***. Deep learning [130] is a hot topic in current research. There are some works focus on deep learning for sequential data (i.e., text, EEG data and DNA), such as Recurrent Neural Networks (RNN) [131], and Long Short Term Memory (LSTM) [132]. Unlike the standard neural network, RNN allows us to operate over sequences of vectors [131]. And RNN uses internal memory to process arbitrary sequences of inputs. LSTM is particularly usable by a category of learning machines, adapted to sequential data [132]. Many deep learning algorithms have been proposed to model sequential data, and then to achieve different learning goals. Despite their flexibility and

power, how to make them parallelized for sequential pattern mining can lead to challenges and open many new research fields.

(6) ***Other important issues***. Beyond the above points, there are some other important issues that can be reasonably considered and further developed, such as visualization techniques and the privacy issue for parallel sequential pattern mining in the big data era. How to design efficient and more flexible PSPM approachs to support iterative and interactive mining is also an interesting issue.

8 Conclusion

Typically, sequential pattern mining algorithms aim at discovering the desired frequent sequential patterns. Since traditional data mining algorithms generally have some problems and challenges while processing large-scale data, parallel data mining has emerged as an important research topic. However, fewer studies summarized the related development in parallel sequential pattern mining in parallel computing environments, or summarized them as a taxonomy structure. In this paper, we first review the key characteristics about some parallelism methods, and then highlight differences and challenges between parallel computing platforms and distributed systems. We then highlight and discuss some related works on parallel sequential pattern mining in several categories. The main contributions are that we investigate recent advances in parallel sequential pattern mining and provide the status of the field in detail, including sequential pattern mining (SPM), parallel frequent itemset mining (PFIM), and parallel sequential pattern mining (PSMP). Both basic algorithms and advanced algorithms for parallel sequential pattern mining are reviewed in several categories, the key ideas, advantages and disadvantages of each approach are also pointed out in details. We further provide some related open-source software of PSPM, that may reduce barriers from research and algorithm implementation. Finally, we briefly point out some challenges and opportunities of parallel sequential pattern mining for future research.

References

[1] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[2] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. IEEE, 1995, pp. 3–14.

[3] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[4] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

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

[6] P. Fournier-Viger, J. C. W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 4, 2017.

[7] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu, "Mining sequential patterns by pattern-growth: The PrefixSpan approach," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.

[8] P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54–77, 2017.

[9] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[10] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on computers*, vol. 100, no. 11, pp. 1025–1034, 1973.

[11] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[12] W. Lee, S. J. Stolfo, and K. W. Mok, "Adaptive intrusion detection: A data mining approach," *Artificial Intelligence Review*, vol. 14, no. 6, pp. 533–567, 2000.

[13] M. S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[14] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 3, p. 9, 2006.

[15] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 721–724.

[16] M. J. Zaki, "Parallel and distributed association mining: A survey," *IEEE concurrency*, vol. 7, no. 4, pp. 14–25, 1999.

[17] W. Gan, J. C. W. Lin, H. C. Chao, and J. Zhan, "Data mining in distributed environment: a survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, 2017.

[18] J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Communications of the ACM*, vol. 53, no. 1, pp. 72–77, 2010.

[19] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, "PFP: parallel fp-growth for query recommendation," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 107–114.

[20] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal, "PARMA: a parallel randomized algorithm for approximate association rules mining in mapreduce," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 85–94.

[21] S. Boggan and D. M. Pressel, "GPUs: an emerging platform for general-purpose computation," ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD COMPUTATIONAL AND INFORMATION SCIENCES DIR, Tech. Rep., 2007.

[22] D. Y. Chiu, Y. H. Wu, and A. L. Chen, "An efficient algorithm for mining frequent sequences by a new strategy without support counting," in *Data Engineering, 2004. Proceedings. 20th International Conference on*. IEEE, 2004, pp. 375–386.

[23] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 429–435.

[24] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[25] C. W. Tsai, C. F. Lai, M. C. Chiang, L. T. Yang *et al.*, "Data mining for internet of things: A survey." *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[26] C. W. Tsai, C. F. Lai, H. C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big Data*, vol. 2, no. 1, p. 21, 2015.

[27] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, P. Michiardi, and F. Pulvirenti, "Pampa-hd: a parallel mapreduce-based frequent pattern miner for high-dimensional data," in *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 839–846.

[28] S. Salah, R. Akbarinia, and F. Masseglia, "A highly scalable parallel algorithm for maximally informative k-itemset mining," *Knowledge and Information Systems*, vol. 50, no. 1, pp. 1–26, 2017.

[29] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li, P. Stolorz, and R. Musick, "Parallel algorithms for discovery of association rules," in *Scalable High Performance Computing for Knowledge Discovery and Data Mining*. Springer, 1997, pp. 5–35.

[30] K. Beedkar and R. Gemulla, "LASH: Large-scale sequence mining with hierarchies," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 491–503.

[31] M. J. Zaki, "Parallel sequence mining on shared-memory machines," *Journal of Parallel and Distributed Computing*, vol. 61, no. 3, pp. 401–426, 2001.

[32] Y. Xu, W. Qu, Z. Li, G. Min, K. Li, and Z. Liu, "Efficient k-means++ approximation with mapreduce," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3135–3144, 2014.

[33] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapReduce," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 674–679.

[34] S. Cong, J. Han, and D. Padua, "Parallel mining of closed sequential patterns," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 562–567.

[35] N. R. Mabroukeh and C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," *ACM Computing Surveys (CSUR)*, vol. 43, no. 1, p. 3, 2010.

[36] C. H. Mooney and J. F. Roddick, "Sequential pattern mining–approaches and algorithms," *ACM Computing Surveys (CSUR)*, vol. 45, no. 2, p. 19, 2013.

[37] Q. Zhao and S. S. Bhowmick, "Sequential pattern mining: A survey," *ITechnical Report CAIS Nayang Technological University Singapore*, vol. 1, p. 26, 2003.

[38] D. C. Anastasiu, J. Iverson, S. Smith, and G. Karypis, "Big data frequent pattern mining," in *Frequent Pattern Mining*. Springer, 2014, pp. 225–259.

[39] J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5342–5359, 2008.

[40] M. Sousa, M. Mattoso, and N. Ebecken, "Data mining on parallel database systems," in *Proc. Intl. Conf. on PDPTA: Special Session on Parallel Data Warehousing, CSREA Press, Las Vegas*, 1998.

[41] N. Khan, I. Yaqoob, I. A. T. Hashem, Z. Inayat, M. Ali, W. Kamaleldin, M. Alam, M. Shiraz, and A. Gani, "Big data: survey, technologies, opportunities, and challenges," *The Scientific World Journal*, vol. 2014, 2014.

[42] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033, 2012.

[43] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[44] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[45] F. Li, B. C. Ooi, M. T. Özsu, and S. Wu, "Distributed data management using mapreduce," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, p. 31, 2014.

[46] M. J. Zaki, "Parallel and distributed data mining: An introduction," in *Large-scale parallel data mining*. Springer, 2000, pp. 1–23.

[47] J. Liu, X. Jin, and Y. Wang, "Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 7, pp. 586–598, 2005.

[48] P. Luo, K. Lü, Z. Shi, and Q. He, "Distributed data mining in grid computing environments," *Future Generation Computer Systems*, vol. 23, no. 1, pp. 84–91, 2007.

[49] W. Rao, L. Chen, A. W. C. Fu, and G. Wang, "Optimal resource placement in structured peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 1011–1026, 2010.

[50] Y. Xue, B. Li, and K. Nahrstedt, "Optimal resource allocation in wireless ad hoc networks: A price-based approach," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 347–364, 2006.

[51] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 24–32, 2013.

[52] Y. Jiang and J. Jiang, "Understanding social networks from a multiagent perspective," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 10, pp. 2743–2759, 2014.

[53] G. S. Almasi and A. Gottlieb, "Highly parallel computing," *Menlo Park, CA (USA)*, 1988.

[54] L. Zeng, L. Li, L. Duan, K. Lu, Z. Shi, M. Wang, W. Wu, and P. Luo, "Distributed data mining: a survey," *Information Technology and Management*, vol. 13, no. 4, pp. 403–409, 2012.

[55] R. Dolbeau, S. Bihan, and F. Bodin, "HMPP: A hybrid multicore parallel programming environment," in *Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007)*, vol. 28, 2007.

[56] P. Fournier-Viger, C. W. Wu, and V. S. Tseng, "Mining maximal sequential patterns without candidate maintenance," in *International Conference on Advanced Data Mining and Applications*. Springer, 2013, pp. 169–180.

[57] S. Hauck and A. DeHon, *Reconfigurable computing: the theory and practice of FPGA-based computation*. Elsevier, 2010, vol. 1.

[58] K. Beedkar, K. Berberich, R. Gemulla, and I. Miliaraki, "Closing the gap: Sequence mining at scale," *ACM Transactions on Database Systems (TODS)*, vol. 40, no. 2, p. 8, 2015.

[59] S. Hong, T. Oguntebi, and K. Olukotun, "Efficient parallel graph exploration on multi-core CPU and GPU," in *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*. IEEE, 2011, pp. 78–88.

[60] C. H. Lin, C. H. Liu, L. S. Chien, and S. C. Chang, "Accelerating pattern matching using a novel parallel algorithm on GPUs," *IEEE Transactions on Computers*, vol. 62, no. 10, pp. 1906–1916, 2013.

[61] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

[62] K. A. Huck and A. D. Malony, "PerfExplorer: A performance data mining framework for large-scale parallel computing," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2005, p. 41.

[63] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data mining and knowledge discovery*, vol. 1, no. 3, pp. 259–289, 1997.

[64] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *proceedings of the 17th international conference on data engineering*, 2001, pp. 215–224.

[65] J. Wang and J. Han, "BIDE: Efficient mining of frequent closed sequences," in *Data Engineering, 2004. Proceedings. 20th International Conference on*. IEEE, 2004, pp. 79–90.

[66] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining: Closed sequential patterns in large datasets," in *Proceedings of the 2003 SIAM international conference on data mining*. SIAM, 2003, pp. 166–177.

[67] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "FreeSpan: frequent pattern-projected sequential pattern mining," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 355–359.

[68] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42, no. 1-2, pp. 31–60, 2001.

[69] F. Masseglia, P. Poncelet, and R. Cicchetti, "An efficient algorithm for web usage mining," *Networking and Information Systems Journal*, vol. 2, no. 5/6, pp. 571–604, 2000.

[70] Z. Yang, Y. Wang, and M. Kitsuregawa, "LAPIN: effective sequential pattern mining algorithms by last position induction for dense databases," in *International Conference on Database systems for advanced applications*. Springer, 2007, pp. 1020–1023.

[71] Z. Yang and M. Kitsuregawa, "LAPIN-SPAM: An improved algorithm for mining sequential pattern," in *Data Engineering Workshops, 2005. 21st International Conference on*. IEEE, 2005, pp. 1222–1222.

[72] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu, "Mining access patterns efficiently from web logs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2000, pp. 396–407.

[73] M. Seno and G. Karypis, "LPMiner: An algorithm for finding frequent itemsets using length-decreasing support constraint," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 505–512.

[74] ——, "SLPMiner: An algorithm for finding frequent sequential patterns using length-decreasing support constraint," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 418–425.

[75] M. El-Sayed, C. Ruiz, and E. A. Rundensteiner, "FS-Miner: efficient and incremental mining of frequent sequence patterns in web logs," in *Proceedings of the 6th annual ACM international workshop on Web information and data management*. ACM, 2004, pp. 128–135.

[76] C. Ezeife, Y. Lu, and Y. Liu, "PLWAP sequential mining: open source code," in *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. ACM, 2005, pp. 26–35.

[77] R. J. Bayardo Jr, "Efficiently mining long patterns from databases," *ACM Sigmod Record*, vol. 27, no. 2, pp. 85–93, 1998.

[78] S. Song, H. Hu, and S. Jin, "Hvsm: a new sequential pattern mining algorithm using bitmap representation," in *International Conference on Advanced Data Mining and Applications*. Springer, 2005, pp. 455–463.

[79] J. Chen, "An updown directed acyclic graph approach for sequential pattern mining," *IEEE transactions on knowledge and data engineering*, vol. 22, no. 7, pp. 913–928, 2010.

[80] P. Fournier-Viger, J. C. W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The SPMF open-source data mining library version 2," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2016, pp. 36–40.

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

[82] C. Kim, J. H. Lim, R. T. Ng, and K. Shim, "Squire: Sequential pattern mining with quantities," *Journal of Systems and Software*, vol. 80, no. 10, pp. 1726–1745, 2007.

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

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

[85] 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, no. 2, pp. 429–463, 2016.

[86] A. Mueller, "Fast sequential and parallel algorithms for association rule mining: A comparison," Tech. Rep., 1998.

[87] J. S. Park, M. S. Chen, and P. S. Yu, "Efficient parallel data mining for association rules," in *Proceedings of the fourth international conference on Information and knowledge management*. ACM, 1995, pp. 31–36.

[88] D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu, "A fast distributed algorithm for mining association rules," in *Parallel and Distributed Information Systems, 1996., Fourth International Conference on*. IEEE, 1996, pp. 31–42.

[89] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li *et al.*, "New algorithms for fast discovery of association rules." in *KDD*, vol. 97, 1997, pp. 283–286.

[90] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," in *Big Data, 2013 IEEE international conference on*. IEEE, 2013, pp. 111–118.

[91] S. Rathee, M. Kaul, and A. Kashyap, "R-Apriori: an efficient Apriori based algorithm on spark," in *Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management*. Acm, 2015, pp. 27–34.

[92] T. Shintani and M. Kitsuregawa, "Mining algorithms for sequential patterns in parallel: Hash based approach," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 1998, pp. 283–294.

[93] M. V. Joshi, G. Karypis, and V. Kumar, "Parallel algorithms for mining sequential associations: Issues and challenges," Technical Report under preparation, Department of Computer Science, University of Minnesota, Minneapolis, Tech. Rep., 1999.

[94] V. Guralnik and G. Karypis, "Parallel tree-projection-based sequence mining algorithms," *Parallel Computing*, vol. 30, no. 4, pp. 443–472, 2004.

[95] A. Demiriz, "webSPADE: A parallel sequence mining algorithm to analyze web log data," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*. IEEE, 2002, pp. 755–758.

[96] X. Yu, J. Liu, X. Liu, C. Ma, and B. Li, "A mapReduce reinforced distributed sequential pattern mining algorithm," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2015, pp. 183–197.

[97] J. W. Huang, S. C. Lin, and M. S. Chen, "DPSP: Distributed progressive sequential pattern mining on the cloud," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 27–34.

[98] S. Qiao, C. Tang, S. Dai, M. Zhu, J. Peng, H. Li, and Y. Ku, "Partspan: Parallel sequence mining of trajectory patterns," in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, vol. 5. IEEE, 2008, pp. 363–367.

[99] C. H. Wu, C. C. Lai, and Y. C. Lo, "An empirical study on mining sequential patterns in a grid computing environment," *Expert Systems with Applications*, vol. 39, no. 5, pp. 5748–5757, 2012.

[100] S. Qiao, T. Li, J. Peng, and J. Qiu, "Parallel sequential pattern mining of massive trajectory data," *International Journal of Computational Intelligence Systems*, vol. 3, no. 3, pp. 343–356, 2010.

[101] I. Miliaraki, K. Berberich, R. Gemulla, and S. Zoupanos, "Mind the gap: Large-scale frequent sequence mining," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 797–808.

[102] C. C. Chen, C. Y. Tseng, and M. S. Chen, "Highly scalable sequential pattern mining based on mapreduce model on the cloud," in *Big Data (BigData Congress), 2013 IEEE International Congress on*. IEEE, 2013, pp. 310–317.

[103] C. C. Chen, H. H. Shuai, and M. S. Chen, "Distributed and scalable sequential pattern mining through stream processing," *Knowledge and Information Systems*, vol. 53, no. 2, pp. 365–390, 2017.

[104] S. Cong, J. Han, J. Hoeflinger, and D. Padua, "A sampling-based framework for parallel data mining," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 2005, pp. 255–265.

[105] D. Yu, W. Wu, S. Zheng, and Z. Zhu, "BIDE-based parallel mining of frequent closed sequences with mapReduce," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2012, pp. 177–186.

[106] V. C. C. Liao and M. S. Chen, "DFSP: a depth-first spelling algorithm for sequential pattern mining of biological sequences," *Knowledge and information systems*, vol. 38, no. 3, pp. 623–639, 2014.

[107] Y. H. Liang and S. Y. Wu, "Sequence-growth: A scalable and effective frequent itemset mining algorithm for big data based on mapreduce framework," in *Big Data (BigData Congress), 2015 IEEE International Congress on*. IEEE, 2015, pp. 393–400.

[108] C. Luo and S. M. Chung, "Parallel mining of maximal sequential patterns using multiple samples," *The Journal of Supercomputing*, vol. 59, no. 2, pp. 852–881, 2012.

[109] K. Wang, Y. Xu, and J. X. Yu, "Scalable sequential pattern mining for biological sequences," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM, 2004, pp. 178–187.

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

[111] J. Ge and Y. Xia, "Distributed sequential pattern mining in large scale uncertain databases," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2016, pp. 17–29.

[112] J. Fowkes and C. Sutton, "A subsequence interleaving model for sequential pattern mining," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 835–844.

[113] M. Sahli, E. Mansour, and P. Kalnis, "Parallel motif extraction from very long sequences," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 549–558.

[114] 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, no. 10, pp. 2645–2657, 2015.

[115] T. P. Hong, C. S. Kuo, and S. C. Chi, "Mining fuzzy sequential patterns from quantitative data," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 3. IEEE, 1999, pp. 962–966.

[116] G.-C. Lan, T.-P. Hong, and H.-Y. Lee, "An efficient approach for finding weighted sequential patterns from sequence databases," *Applied intelligence*, vol. 41, no. 2, pp. 439–452, 2014.

[117] G. Ruan, H. Zhang, and B. Plale, "Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 32–39.

[118] M. Zihayat, Z. Z. Hut, A. An, and Y. Hut, "Distributed and parallel high utility sequential pattern mining," in *Big Data (Big*

Data), 2016 IEEE International Conference on. IEEE, 2016, pp. 853–862.
[119] M. Muzammal and R. Raman, "Mining sequential patterns from probabilistic databases," *Knowledge and Information Systems*, vol. 44, no. 2, pp. 325–358, 2015.
[120] M. Hassani and T. Seidl, "Towards a mobile health context prediction: Sequential pattern mining in multiple streams," in *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, vol. 2. IEEE, 2011, pp. 55–57.
[121] M. B. Harries, C. Sammut, and K. Horn, "Extracting hidden context," *Machine learning*, vol. 32, no. 2, pp. 101–126, 1998.
[122] S. Memeti and S. Pllana, "Accelerating dna sequence analysis using intel (r) xeon phi (tm)," in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 3. IEEE, 2015, pp. 222–227.
[123] K. Wang, E. Sadredini, and K. Skadron, "Sequential pattern mining with the micron automata processor," in *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 2016, pp. 135–144.
[124] H. Cheng, X. Yan, and J. Han, "IncSpan: incremental mining of sequential patterns in large database," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 527–532.
[125] B. Zhang, C. W. Lin, W. Gan, and T. P. Hong, "Maintaining the discovered sequential patterns for sequence insertion in dynamic databases," *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 131–142, 2014.
[126] J. W. Huang, C. Y. Tseng, J. C. Ou, and M. S. Chen, "A general model for sequential pattern mining with a progressive database," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 9, pp. 1153–1167, 2008.
[127] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, "Differentially private data publishing and analysis: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1619–1638, 2017.
[128] N. Tatti and J. Vreeken, "The long and the short of it: summarising event sequences with serial episodes," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 462–470.
[129] A. Bhattacharyya and J. Vreeken, "Efficiently summarising event sequences with rich interleaving patterns," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 795–803.
[130] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.
[131] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
[132] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Wensheng Gan is currently a joint Ph.D. candidate at the Department of Computer Science, University of Illinois at Chicago, USA; and a Ph.D. candidate at the School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Guangdong, China. He received B.S. and M.S. degrees in Computer Science from South China Normal University and Harbin Institute of Technology (Shenzhen), Guangdong, China, in 2013 and 2015, respectively. His research interests include data mining, utility mining, pattern discovery, and Big Data technologies. He has published more than 50 research papers in peer-reviewed journals and international conferences.

Jerry Chun-Wei Lin received his Ph.D. in Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2010. He is now working as an associate professor at the Harbin Institute of Technology (Shenzhen), Shenzhen, China. His research interests include data mining, privacy-preserving and security, Big Data analytics, and social networks. He has published more than 200 research papers in peer-reviewed international conferences and journals, which have received more than 1900 citations. He is the co-leader of the popular SPMF open-source data-mining library and the Editor-in-Chief of *Data Mining and Pattern Recognition* (DSPR), Associate Editor of *Journal of Internet Technology*, and a member of the Editorial Board of *Intelligent Data Analysis*.

Philippe Fournier-Viger is a full professor and Youth 1000 scholar at the Harbin Institute of Technology (Shenzhen), Shenzhen, China. He received a Ph.D. in Computer Science from the University of Quebec, Montreal, in 2010. His research interests include data mining, pattern mining, sequence analysis and prediction, text mining, e-learning, and social network mining. He has published more than 200 research papers in peer-reviewed international conferences and journals, which have received more than 2200 citations. He is the founder of the popular SPMF open-source data-mining library, which has been cited in more than 700 research papers since 2010. He is also Editor-in-Chief of *Data Mining and Pattern Recognition* (DSPR).

Han-Chieh Chao has been the president of National Dong Hwa University since February 2016. He received M.S. and Ph.D. degrees in Electrical Engineering from Purdue University in 1989 and 1993, respectively. His research interests include high-speed networks, wireless networks, IPv6-based networks, and artificial intelligence. He has published nearly 350 peer-reviewed professional research papers. He is the Editor-in-Chief (EiC) of IET Networks' *Journal of Internet Technology*. He is the founding EiC of the *International Journal of Internet Protocol Technology* and the *International Journal of Ad Hoc and Ubiquitous Computing*. Dr. Chao has served as a guest editor for ACM MONET, IEEE JSAC, *IEEE Communications Magazine*, *IEEE Systems Journal*, *Computer Communications*, *IEEE Proceedings Communications*, *Wireless Personal Communications*, and *Wireless Communications & Mobile Computing*. He is an IEEE Senior Member and a fellow of IET.

Philip S. Yu received a B.S. degree in electrical engineering from National Taiwan University, M.S. and Ph.D. degrees in electrical engineering from Stanford University, and an MBA from New York University. He is a distinguished professor of computer science with the University of Illinois at Chicago (UIC) and also holds the Wexler Chair in Information Technology at UIC. Before joining UIC, he was with IBM, where he was manager of the Software Tools and Techniques Department at the Thomas J. Watson Research Center. His research interests include data mining, data streams, databases, and privacy. He has published more than 1,000 papers in peer-reviewed journals (i.e., TKDE, TKDD, VLDB J., and ACM TIST) and conferences (KDD, ICDE, WWW, AAAI, SIGIR, ICML, CIKM, etc). He holds or has applied for more than 300 U.S. patents. Dr. Yu is the Editor-in-Chief of *ACM Transactions on Knowledge Discovery from Data*. He received the ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of Big Data, and the IEEE Computer Society 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining, and anonymization of Big Data. Dr. Yu is a fellow of the ACM and the IEEE.