A Survey of Parallel Sequential Pattern Mining

WENSHENG GAN, Harbin Institute of Technology (Shenzhen)
JERRY CHUN-WEI LIN, Harbin Institute of Technology (Shenzhen) and Western Norway University of Applied Sciences
PHILIPPE FOURNIER-VIGER, Harbin Institute of Technology (Shenzhen)
HAN-CHIEH CHAO, National Dong Hwa University
PHILIP S. YU, University of Illinons at Chicago

With the growing popularity of 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. As a fundamental task of data mining, sequential pattern mining (SPM) is used in a wide variety of real-life applications. However, it is more complex and challenging than other pattern mining tasks, i.e., frequent itemset mining and association rule mining, and also suffers from the above challenges when handling the large-scale data. To solve these problems, mining sequential patterns in a parallel or distributed computing environment has emerged as an important issue with many applications. In this article, an in-depth survey of the current status of parallel SPM (PSPM) is investigated and provided, including detailed categorization of traditional serial SPM approaches, and state-of-the art PSPM. We review the related work of PSPM in details 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, disadvantages, and summarization) of these parallel approaches of PSPM. Some advanced topics for PSPM, including parallel quantitative/weighted/utility SPM, PSPM from uncertain data and stream data, hardware acceleration for PSPM, are further reviewed in details. Besides, we review and provide some well-known open-source software of PSPM. Finally, we summarize some challenges and opportunities of PSPM in the big data era.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; • **Mathematics of computing** → *Combinatorics*; • **Information systems** → **Data mining**;

Additional Key Words and Phrases: Data science, big data, data mining, parallelism, sequential pattern

This research was partially supported by the Shenzhen Technical Project under KQJSCX20170726103424709 and JCYJ20170307151733005.

Authors' addresses: W. Gan and P. Fournier-Viger, Harbin Institute of Technology (Shenzhen), HIT Campus of University Town of Shenzhen, Shenzhen 518055, China; emails: wsgan001@gmail.com, philfv@hitsz.edu.cn; J. C.-W. Lin (corresponding author), Harbin Institute of Technology (Shenzhen), Shenzhen, China and Western Norway University of Applied Sciences, Inndalsveien 28, 5063 Bergen, Norway; email: jerrylin@ieee.org; H.-C. Chao, National Dong Hwa University, No. 1, Sec. 2, Da Hsueh Rd., Shoufeng, Hualien 97401, Taiwan; email: hcc@ndhu.edu.tw; P. S. Yu, University of Illinons at Chicago, 1200 W Harrison St, Chicago, IL 60607, USA; email: psyu@uic.edu.

1556-4681/2019/06-ART25 $15.00
https://doi.org/10.1145/3314107

ACM Reference format:
Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, and Philip S. Yu. 2019. A Survey of Parallel Sequential Pattern Mining. *ACM Trans. Knowl. Discov. Data* 13, 3, Article 25 (June 2019), 34 pages.
https://doi.org/10.1145/3314107

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 [1–3, 52, 112]. KDD has numerous real-life applications and is crucial to some of the most fundamental tasks such as frequent itemset mining (FIM) [34, 52], association rule mining (ARM) [1, 138], sequential pattern mining (SPM) [33, 95, 112, 119, 134], clustering [14, 63], classification [93, 99], and outline detection [71].

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, 14, 21, 46, 52, 128]. 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 demand 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) [40, 136] is a type of computing architecture in which several processors execute or process an application. Research and development work in the area of PDM 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. multi-core 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 [27] and achieved certain speedup compared with single-node methods. However, some previous studies [27, 73, 103] show that the MapReduce framework is not suitable for FIM 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 [3] which was proposed by Agrawal and Srikant. It is designed for finding frequent itemsets (FIs) and then extracting the association rules. Note that FIs 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

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*

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*

analysis. Some pattern mining techniques, such as FIM [3, 52] and ARM [3], 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, deoxyribonucleic acid (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), occurred 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 times. 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.

To address this issue, another concept of pattern mining named 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* (FS) 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, SPM is used in a wide variety

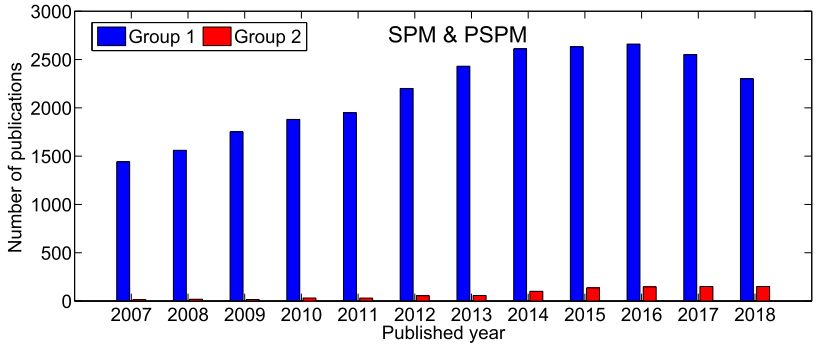

Fig. 1. Number of published papers that use "parallel sequential pattern mining" in subareas of data mining and big data. These publication statistics are obtained from *Google Scholar*; the search phrase of *Group 1* and *Group 2* is defined as the subfield named with the exact phrase "parallel sequential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

of real-life applications, such as market-basket analysis [2], web mining [16], bioinformatics [24], classification [9], and finding copy–paste bugs in large-scale software code [17].

In the past two decades, pattern mining (i.e., FIM, ARM, and SPM) has been extensively studied and successfully applied in many fields [33, 34]. Meanwhile, to meet the demand of large-scale and high-performance computing, as mentioned before, PDM has received considerable attention over the past decades [40, 117, 118, 136], including parallel FIM (PFIM) [8, 73, 106], parallel ARM (PARM) [103, 142], parallel SPM (PSPM) [12, 139], parallel clustering [126, 146], 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 [24], it requires a truly parallel computing on massive large-scale DNA. On the one hand, the serial SPM 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 [26]. Thus, solutions of the SPM task that scale are quite important. Up to now, the problem of PSPM has attracted a lot of attention [12, 139]. Figure 1 shows the number of published papers from 2007 to 2017 where *Group 1* denotes the search keywords of "SPM"/"sequence mining," and *Group 2* denotes the search keywords of "PSPM"/"parallel sequence mining"/MapReduce "sequential pattern." Figure 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 SPM have been previously studied [84, 90, 145]. Recently, Fournier-Viger et al. published an up-to-date survey of the current status of serial SPM [33]. There is, unfortunately, no survey of PSPM methods yet. In 2014, Anastasiu et al. published an article on the big data frequent pattern mining [6], which describes several serial and parallel approaches of scalable FIM, sequence mining, and frequent graph mining. However, this article is not specific for PSPM, 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 10 years of theoretical development of big data, a significant number of new technologies and applications have appeared in this area. Thus, we review and summarize current status (i.e., the big data platforms and tools, the new technologies, application, and advanced topics) of PSPM in details.

The question then posed is how can one best summarize the related studies in various types of PSPM in parallel computing environments and make a general taxonomy of them? The methods summarized in this article not only fit for parallel computing [7, 16, 111], but are also good references for other related work, such as data mining [21], big data technologies [66, 68, 117, 118, 125], distributed systems [17, 113], and database management [72]. Thus, this article aims at reviewing the current status of PSPM, and providing in-depth descriptions on a number of representative algorithms. The main contributions of this article are described below.

- —We review some related works on serial SPM 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 PSPM on distributed environments in recent years. This is a high-level survey about parallel techniques for SPM 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 PSPM).
- —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 PSPM task for future research are briefly summarized and discussed.

The rest of this article 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 of SPM. Section 4 highlights and discusses the state-of-the-art research efforts on different approaches, under four taxonomies, for PSPM. Some advanced topics for PSPM and the related open-source software are further reviewed in Sections 5 and 6, respectively. Section 7 briefly summarizes some challenges and opportunities in PSPM. 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 [136, 137]. 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 [136, 137]. 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 [136, 137], there are some well-known methods of parallelism: task parallelism (divide and conquer strategy and 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, i.e., 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, i.e., record-based data parallelism and attribute-based data parallelism. In [136, 137], Zaki had pointed that the records or attribute lists in record-based data parallelism are horizontally partitioned among the processors. In contrast, the attributed in attributed-based data parallelism are divided, thus each processor is responsible for an equal number of attributes.
(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 [136, 137]:

—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 [17, 113]. 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 [17, 113], there are two aspects of distributed systems: (1) independent computing elements and (2) single system w.r.t. middleware. There are some important features of distributed systems, including (1) concurrency, multiprocess 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 [17, 113]. There are many types of distributed systems, such as grids [80, 83], peer-to-peer (P2P) systems [101], ad-hoc networks [127], cloud computing systems [47], and online social network systems [64]. 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 [5]. 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.

[1] http://en.wikipedia.org/wiki/Task_parallelism.
[2] http://en.wikipedia.org/wiki/Data_parallelism.
[3] https://en.wikipedia.org/wiki/Parallel_computing.

Both distributed mining and parallel mining can speedup the data mining process. However, they have several differences. In 1999, Zaki pointed out that there are some differences between parallel and distributed ARM [136]. 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 ARM) [136]. 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 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 [143]. 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 PDM 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 [29, 100], grid computing [80, 83], graphics processing units (GPUs) [7, 16], cloud computing [47], 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 [36], field-programmable gate array (FPGA) [55], cluster architecture [5], or the massive parallelism of GPUs [11]. Some PDM 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 [29]. Recently, an SPM model using multi-core processors is presented in [62].
(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.
(3) *Reconfigurable computing with FPGA:* A FPGA is, in essence, a computer chip that can rewire itself for a given task [55]. A new computing paradigm, reconfigurable computing, has received wide attention and is the focus of extensive research. Reconfigurable computing uses an 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) *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 GPUs. GPUs are co-processors that have been heavily optimized for computer graphics processing [7, 16]. Recently, some GPU-based tools are

[4] http://hadoop.apache.org.
[5] https://en.wikipedia.org/wiki/Multi-core_processor.
[6] https://en.wikipedia.org/wiki/Grid_computing.
[7] https://en.wikipedia.org/wiki/Graphics_processing_unit.

developed, such as a parallel graph exploration system on multi-core CPU and GPU [57], and a novel parallel algorithm on GPUs to accelerate pattern matching [76].

(5) *MapReduce* [27]: 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 [27] 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* [135]: 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 speedup computation for iterative algorithms such as the apriori algorithm and some other machine learning algorithms. In general, spark [135] can be 1–2 orders of magnitude faster than MapReduce [27].

(7) *PerfExplorer* [61]: It is a computing framework for large-scale PDM. 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 [61].

3 STATE-OF-THE-ART SERIAL SEQUENTIAL PATTERN MINING

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

3.1 Frequent Itemset Mining Versus Sequential Pattern Mining

First, let us introduce some preliminaries and the problem statement related to 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 *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 *FI* in a database if its support is no less than the user-defined minimum support threshold (*minsup*). Therefore, the problem of FIM is to discover all itemsets which have a support no less than the defined minimum support threshold, that is $sup(X) \geq minsup$ [3]. As the most important mining task for a wide range of real-world applications, FIM or ARM has attracted a lot of attention [1, 3, 21, 34, 40, 46, 52].

In 1995, Srikant and Agrawal first extended the concept of the FIM model to handle sequences [2]. They first introduced and formulated the SPM problem. Different from FIM, SPM aims at discovering frequent subsequences as interesting patterns in a sequence database which contains

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	$<\{a,b\},\{c\},\{d,f,g\},\{g\},\{e\}>$

the embedded time-stamp information of event. According to the definitions from Mannila et al. [85], 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 *SID*) 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 $k = |A_1| + |A_2| + \cdots + |A_n|$. For example, as shown in Table 3, the sequence $<\{a,d\},\{c\},\{d,g\},\{g\},\{e\}>$ is a seven-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 S_β is called a *super-sequence* of S_α if there exists an integer $1 < i_1 < \cdots < i_n < m$ 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, SPM is more complex and challenging than FIM. There are some important reasons for this. First, due to the absence of time constraints in FIM not presenting in SPM, SPM has a potentially huge set of candidate sequences [51]. 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 $2^{100} - 1 \approx 10^{30}$. Second, there are further difficulties mining longer sequential patterns [51], 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 [33], such as web mining [16], classification [9], and mining motifs from biological sequences [24]. Some well-known serial algorithms for SPM, such as AprioriAll [2], GSP [112], BIDE [120], CloSpan [129], FreeSpan [50], PrefixSpan [51], SPADE [140], and so on, have been proposed. Since many parallel SPM algorithms are extended by the traditional serial SPM approaches, it is quite necessary for us to have

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 [112]	Generalized sequential patterns (max/min gap, window, and taxonomies).	1996
PSP [86]	Retrieval optimizations and more efficient than GSP.	1998
SPADE [140]	Sequential pattern discovery using equivalence classes.	2001
SPAM [9]	Sequential pattern mining with bitmap representation.	2002
LAPIN [131]	SPM with last position induction, which is categorized into two classes, LAPIN-LCI and LAPIN-Suffix.	2004
LAPIN-SPAM [130]	Last position induction sequential pattern mining.	2005

an understanding of the systematic characteristics of traditional serial SPM 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 4–7, respectively.

Apriori-based techniques: In 1995, Agrawal and Srikant first introduced a 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 non-empty subsets of a FI must also be frequent" [3]. Therefore, AprioriAll holds the *downward closed property* (also called *anti-monotonic*), and utilizes this anti-monotonic to prune the search space. In general, the following two properties are often utilized in SPM to speedup computation:

— *Subsequence-infrequency-based pruning*: Any supersequence 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 FS is also frequent, thus it can be safely pruned from the set of candidates [10]. For example, if sequence $<a, b, d>$ is frequent and occurs eight times, then any its subsequence ($<a>$, $<b>$, $<d>$, $<a, b>$, $<a, d>$, and $<b, b>$) is frequent and occurs at least eight 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 [112]. 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 [86] 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 [112] 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 [140]. 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) [131]. They further developed the LAPIN-SPAM [130] 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 the following three features:

— *BFS*: Apriori-based algorithms commonly use the BFS (w.r.t. level-wise). They construct all k-sequences together in each kth iteration while exploring the search space.
— *Generate-and-test*: This feature is introduced by Agrawal et al. in the apriori algorithm [3]. It is used in the early SPM 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 SPM algorithm named FreeSpan was introduced by Han et al. [50]. By using a frequent item list (f-list) and S-matrix, it obtains all FS 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 [7, 50]. The projected database can significantly reduce the size of databases as frequent patterns are considered by the depth-first search.

The web access pattern (WAP)-mine algorithm, which utilizes a WAP-tree, was proposed by Pei et al. [96]. 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 [51]. 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 FS, sequential patterns can be recursively grown in each projected subdatabase. Thus, PrefixSpan exhibits better performance than both GSP [112] and FreeSpan [50]. However, when dealing with large dense databases that have large itemsets, it is worse than that of SPADE [140]. As shown in Table 5, some other pattern growth algorithms for SPM have been developed, such as LPMiner [107], SLPMiner [108], WAP-mine [96], and FS-Miner [30]. With consideration of length decreasing support, the LPMiner [107] and SLPMiner [108] algorithms are introduced. In 2005, Ezeife et al. proposed a pattern-growth and early pruning hybrid method named PLWAP [31]. PLWAP utilizes a binary code assignment method to construct a preordered, linked, position-coded WAP-tree. Compared to the WAP algorithm, PLWAP can significantly reduce runtime and provide a position code mechanism. Inspired by FP-tree [52] and the projection technique for quickly mining sequential patterns [7], a tree algorithm named FS-Miner is developed [30]. FS-Miner resembles WAP-mine [96]. Moreover, it supports incremental mining and interactive mining [32].

Early-pruning techniques: Although the projection-based approaches (i.e., FreeSpan and PrefixSpan) can achieve a significant improvement over apriori-based approaches, the projection

Table 5. Pattern Growth Algorithms for Sequential Pattern Mining

Name	Description	Year
FreeSpan [50]	Frequent pattern-projected sequential pattern mining.	2000
WAP-mine [96]	SPM with suffix growth.	2000
PrefixSpan [51]	PREFIX-projected sequential pattern mining.	2001
LPMiner [107]	Sequential pattern mining with length decreasing support.	2001
SLPMiner [108]	Sequential pattern mining with length decreasing support.	2002
FS-Miner [30]	SPM with suffix growth.	2004
LAPIN-Suffix [131]	SPM with suffix growth.	2004
PLWAP [31]	SPM with prefix growth.	2005

Table 6. Algorithms for Sequential Pattern Mining with Early Pruning

Name	Description	Year
HVSM [110]	Bitwise operation and position induction	2005
LAPIN-SPAM [130]	Bitwise operation	2005
PLWAP [31]	Position induction	2005
LAPIN [131]	Position induction	2007
DISC-all [24]	Position induction and prefix growth	2007
UDDAG [20]	Up-down directed acyclic graph for sequential pattern mining	2010

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 SPM algorithm (named HVSM for short) [110] is developed using bitmap representation. Instead of using candidate generate-and-test or the tree projection, the LAPIN algorithm [131] 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 [24]. 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 [20] 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. In other perspectives, SPM with regular expression-based constraints [116], SPIRIT algorithm [43], and SPM with declarative constraints [13] have been extensively studied.

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

Table 7. Constraint-based Algorithms for Sequential Pattern Mining

Name	Description	Constraint	Character	Year
CloSpan [129]	The first algorithm for mining closed sequential patterns.	—Closed	—Based on the GSP algorithm.	2003
BIDE [120]	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 [67]	Sequential pattern mining with quantities, Apriori-QSP and PrefixSpan-QSP.	—Pattern with quantities	—Apriori based and PrefixSpan based.	2007
ClaSP [48]	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 [36]	Maximal sequential pattern miner without candidate maintenance.	—Maximal	—It neither produces redundant candidates nor stores intermediate candidates in main memory.	2013
VMSP [35]	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 [38]	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

Pattern miner without candidate maintenance) was proposed [36]. The maximal representation may cause the information loss of support, thus another condense representation named closed pattern was introduced [129]. Up to now, some algorithms for mining closed sequential patterns have been proposed, such as CloSpan [129], BIDE [120], ClaSP [48], CloFS-DBV [115], and CloFAST [38]. CloSpan [129] 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 [120]. 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 [48]. Recently, a more fast algorithm called CloFAST was proposed for mining closed sequential patterns using sparse and vertical id-lists [38]. 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 SPM 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 [84, 90, 145], and an up-to-date survey of the current status of sequent pattern mining can be referred to [33].

4 STATE-OF-THE-ART PARALLEL SEQUENTIAL PATTERN MINING

In this section, we first briefly overview the current status of PFIM. We also cover some special categories of PSPM 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, SPM is highly related to FIM, 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 PFIM. The problem of FIM in parallel environments has been extensively studied so far, and a number of approaches have been explored to address this problem, such as Parallel Efficient Association Rules (PEAR) [91], Parallel Partition Association Rules (PPAR) [91], PDM [94], parallel-based Eclat (ParEclat) [142], PaMPa-HD [8], PHIKS [106], and so on. In 1995, Mueller first proposed two parallel algorithms, PEAR [91] and PPAR [91]. Cheung et al. also proposed an algorithm named PDM to parallel mining of association rules [94], and the Fast Distributed Mining algorithm for distributed databases [23]. An adaptation of the FP-growth algorithm to MapReduce [27] called PFP (parallel FP-tree) is presented in [73]. 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 [141], the ParEclat [142], and the distributed Eclat [89] 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 [27] or spark [135] environment. Some results of these efforts are PARMA [103], a PFIM algorithm with spark (R-apriori) [102], PaMPa-HD [8], and PHIKS [106]. PARMA [103] 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 FIM. Up to now, large numbers of studies of PFIM have been extensively studied compared to PSPM. The development of PFIM is still an active area in research, where the up-to-date advances of PFIM can be referred to [6, 40]. 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 PFIM, a variety of mining algorithms has been developed for 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, PSPM has received much attention.

First, the partition-based parallel methods for SPM are introduced and reviewed. The central idea behind partition-based algorithms is described below. Here, the partition mainly in terms of the computation partition and data partition. For example, the database itself can be shared, partially or totally replicated, or partitioned among the available distributed nodes [137]. Notice that the partition can be achieved by using round-robin, hash, or range scheduling. As shown in Table 8, some partition-based parallel methods for SPM have been developed. A comprehensive working example of partition-based SPM algorithm with task parallelism is illustrated below, to show how the problem of PSPM is solved. It could help to understand the idea of PSPM.

Note that Figure 2 is an illustrated example of the SPSPM algorithm (simply partitioned SPM with task parallelism) [109]. Assuming there are three nodes in the mining system, denoted as

Table 8. Partition-based Algorithms for Parallel Sequential Pattern Mining

Name	Description	Pros.	Cons.	Year
HPSPM [109]	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 [109]	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 [109]	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 [65]	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 [65]	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 [49]	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

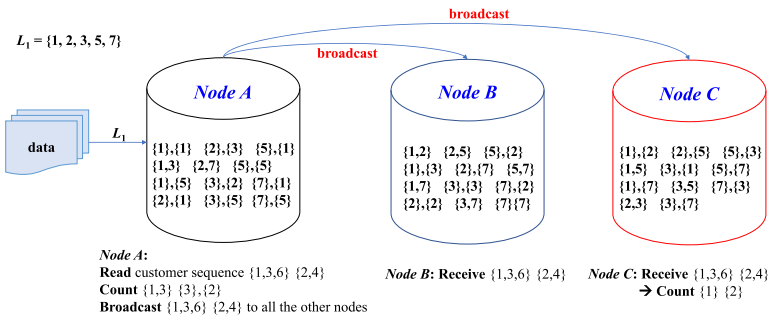

Fig. 2. An illustrated example of the SPSPM algorithm [109].

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 Figure 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 first 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 [109]. Based on the classical GSP approach [112], they proposed three parallel approaches [109]: NPSPM (non-partitioned SPM), SPSPM (simply partitioned SPM), and HPSPM (hash partitioned SPM). 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 [109]. Although HPSPM has the best performance among the three variants, it also suffers, however, several limitations. For example, when the object timelines are very large, or when the counting method for generalized SPM is used, HPSPM is very time-consuming and memory cost.

EVE and EVECAN: Similar assumptions of NPSPM were made in EVE [65] and EVECAN [65]. 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, 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.

Data parallel formulation (DPF), static task-parallel formulation (STPF), and dynamic task-parallel formulation (DTPF): Guralnik and Karypis developed three tree-projection-based parallel sequence mining algorithm [49], named DPF, STPF, and DTPF, for distributed-memory architectures. These projection algorithms are parallelized using the DPF and the task parallel formulation (TPF). They are intrinsically similar to the PrefixSpan algorithm [51], 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 Parallel mining of Closed Sequential Patterns (Par-CSP) algorithm [26]. 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

Table 9. Apriori-based Algorithms for Parallel Sequential Pattern Mining

Name	Description	Pros.	Cons.	Year
pSPADE [139]	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 [28]	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 an apriori-based, vertical formatting method similar to SPADE, it needs several partial scans of the database.	2002
DGSP [133]	A distributed GSP algorithm based on MapReduce.	—DGSP optimizes the workload balance, partitions the database, and assigns the fragments to Map workers.	—It may perform 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 [59]	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 [124]	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 [19]	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 [18]	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

apriori-based algorithms for PSPM are introduced. Moreover, the key differences, advantages, and disadvantages between these approaches are highlighted in Table 9.

pSPADE and webSPADE: Zaki first extended the efficient SPADE algorithm to the shared memory parallel architecture, called pSPADE [139]. 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 TPF approach (Recursive Dynamic Load Balancing, RDLB), and uses two DPFs (single versus 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 [28]. 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 [27] framework named DGSP was introduced in [133]. 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 straightforward 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) [59]. 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 [124], were proposed to find trajectory patterns by extending the traditional GSP algorithm [112]. 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 [124]. 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 to count the relative occurrence frequencies of candidate sequences. Besides, GridGSP uses the Count Distribution 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 [97] and MG-FSM [88]. 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.

SPAM algorithm on the Cloud (SPAMC) and SPAMC-UDLT: Recently, Huang et al. extend the classic SPAM algorithm to a MapReduce version, named as SPAMC. Cloud-based SPAMC utilizes an iterative MapReduce to quickly generate and prune candidate sequences while constructing the lexical sequence tree [19]. 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 [18]. SPAMC-UDLT dramatically improves overall performance without launching multiple MapReduce rounds, and provides perfect load balancing across machines in the cloud [18]. The results [18] 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 PSPM. In general, all these algorithms may also suffer

Table 10. Pattern Growth Algorithms for Parallel Sequential Pattern Mining

Name	Description	Pros.	Cons.	Year
Par-ASP [25]	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 [26]	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 [97]	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 [132]	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 [75]	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 [74]	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

from some drawbacks of serial apriori-based SPM algorithm. Thus, they do not scale well and cannot 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 SPM 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 [25] and Sequence-Growth [74].

PFP, Parallel PrefixSpan to mine all sequential-patterns (Par-ASP), and Par-CSP: In order to balance mining tasks, Cong et al. designed several pattern-growth models, PFP-growth [25], Par-ASP [25], and Par-CSP [26], 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 [51], 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 subsequences are done while estimating their execution times. After that, task distribution is done in the same way as in the Par-FP algorithm [25]. 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 SPM has yet been proposed. In 2005, Cong et al. first proposed the Par-CSP (Parallel Closed SPM) algorithm [26] based on the BIDE algorithm. Par-CSP [26] 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 speedup 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 [82].

In addition, sequential mining can be applied into biological sequence mining, such as a protein DNA analysis. Recently, the 2PDF-Index [122], 2PDF-Compression [122], and DFSP [75] algorithms were proposed and applied to scalable mining sequential patterns for biological sequences. DFSP [75] 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 and DNA sequences), while SPAMC-UDLT [18] 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 the early technologies, including the partition-based and apriori-based algorithms for PSPM. Moreover, the key differences, advantages, and disadvantages between these approaches are highlighted in Table 9.

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 [88] and its enhanced version MG-FSM+ [11] are the scalable distributed (i.e., shared-nothing) algorithms for gap-constrained SPM on MapReduce [27]. 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 [88] and MG-FSM+ [11] 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 [12]. Inspired by MG-FSM, LASH first partitions

Table 11. Hybrid Algorithms for Parallel Sequential Pattern Mining

Name	Description	Pros.	Cons.	Year
MG-FSM [88]	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 [75]	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 [45]	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+ [11]	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 [12]	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 [44]	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) [37]	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

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 [27], 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 [105], 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, cache aware search space traversal model and 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 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, which usually have a significant performance improvement than the above partition-based, apriori-based, and pattern growth algorithms. 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 PSPM are provided and discussed in details, including parallel quantitative/weighted/utility SPM, PSPM from uncertain data and stream data, hardware accelerator for PSPM. For specific problems and tasks in real-world applications, these advanced algorithms can provide more choices and options to user.

When someone is beginning his/her way in data mining, especially in SPM, he/she often face the problem of choosing the most appropriate algorithm for his/her specific task. There is no doubt that no one solution or one approach that fits all. In general, some problems may be very specific and require a unique approach. In the above sections, we have provided a detailed survey and discussion for the different categories of PSPM so far. In our opinions, people who want to use the PSPM algorithm should first understand their data, then categorize the problem or task, and understand the specific constraints and the requirements/goals (e.g., business requirement, accuracy, efficiency, maximal memory usage, and scalability). Finally, they can find the available algorithms for his/her specific requirement.

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 [58], weight [69, 77, 134], or utility with different items/objects [41, 42, 79]. 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 SPM. Up to now, some approaches have been proposed to handle quantity, weight, or utility constraints in SPM, and a literature review has been provided by Gan et al. [42]. Many sequential pattern algorithms have been introduced, such as quantitative sequential pattern [58, 67], weighted sequential pattern [69, 134], utility-driven sequential pattern [4, 42], and so on. All of these efforts rely on sequence data 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 [104] for parallel and quantitative mining of sequential patterns at scale, and the spark-based BigHUSP algorithm [148] for

mining high-utility sequential patterns. As one of the big data models for utility mining [42], BigHUSP is designed based on the Apache spark platform [135] and takes advantage of several merit properties of spark such as distributed in memory processing, fault recovery, and high scalability [148]. On the one hand, both PESMiner [104] and BigHUSP [148] 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 [27] or the Apache spark [135] 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 [92], while traditional data mining (i.e., FIM, ARM, and SPM) algorithms are inapplicable when handling uncertain data. Other related algorithms for PSPM are still being developed, such as the PSPM of large-scale databases by using a probabilistic model [37]. 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 [45]. An uncertain sequence database $D = \{d_1, \ldots, d_n\}$ is abstracted by a Resilient Distributed Datasets (RDD) [135] 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 [92], it is challenging in terms of efficiency and scalability. Ge and Xia further developed a Distributed Sequential Pattern mining algorithm in large-scale uncertain databases based on spark, which relies on a memory-efficient distributed dynamic programming (DP) approach [44]. Although MapReduce [27] 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 [44]. For bioinformatics, a new spark-based framework was proposed to mine sequential patterns from uncertain big DNA [54].

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 [53]. Few stream algorithms have been developed for parallel mining sequential patterns up till now. Recently, Chen et al. proposed two PSPM algorithms, SPAMC [19] and SPAMC-UDLT [18], for processing stream data. Since SPAMC is still not scalable enough for mining large-scale data, the SPAMC-UDLT algorithm [18] 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

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

can effectively generate all the frequent sequential patterns by using stream mining operation. In cases of multiple streams in parallel, the MSSBE algorithm [54] 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 speedup 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, GPUs [7, 16] 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 [76]. Based on the traditional AC algorithm, they implement three CPU versions and one GPU version as follows: ACCPU, DPACOMP, PFACOMP, and PFACGPU [76].

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) [87]. Recently, Wang et al. proposed a hardware-accelerated solution, Automata Processor (AP)-SPM and GPU-GSP, for SPM [121]. The major ideas are summarized as follows: It uses Micron's new 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 [121], it has been shown that the AP-accelerated solution outperforms PrefixSpan [51] and SPADE [140] on multi-core CPU, and scales well for larger datasets.

6 OPEN-SOURCE SOFTWARE

Although the problem of SPM has been studied for more than two decades, and the advanced topic of parallel SPM 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 and 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 [9] 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 [32] 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 SPM. SPMF has the largest collection of implementations of various algorithms for SPM. It

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 [9]	Gehrke et al.	http://himalaya-tools.sourceforge.net/Spam
SPMF [112]	Fournier-Viger et al.	http://www.philippe-fournier-viger.com/spmf
MLlib	Apache spark	https://spark.apache.org/mllib/
MG-FSM [86]	Miliaraki et al.	https://github.com/uma-pi1/mgfsm
LASH [12]	Beedkar et al.	http://uma-pi1.github.io/lash

provides the Java code for AprioriAll [2], GSP [112], PrefixSpan [51], SPADE [140], SPAM [9], and many others. The only problem is that SPMF does not provide any parallel algorithms.

— *Machine Learning Library (MLlib):* MLlib is a scalable Apache-spark-based MLlib. MLlib contains many algorithms and utilities, including FIM, ARM, SPM, as well as many other algorithms for other mining tasks. Algorithms in MLlib are written in Java, Scala, Python, and R. Up to now, MLlib has become an important and widely used framework with SPM algorithms.

— *MG-FSM and MG-FSM+:* MG-FSM [88] 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 [12] 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 [68]. Although useful developments have been made in the field of parallel computing platforms, there are still some technical challenges [17, 113]. Despite advances in the field of PDM, especially PFIM and PSPM, both in academia and industry, significant challenges still remain in PSPM. Some challenges which need to be dealt with for realizing PSPM are, respectively, explored in the following section.

(1) *Complex types of sequence data:* Sequential data commonly occurs in many fields, current developments in PSPM have successfully addressed some types of sequence data, including the general type of sequence, sequence data containing quantity/weight/utility [42], uncertain sequential data [45], stream data [18], 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 example, the spatio-temporal sequential data contains rich semantic features and information (i.e., time-space information, location, spatial, and temporal relationships among data points

of trajectories) than the traditional sequential data. How to improve the mining efficiency, and how to extract more rich information rather than the support-based occurrence are more interesting and helpful to real-world 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 multi-modality 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*. How to integrate different knowledge from the *diversity* of multi-modal data using PSPM algorithms is a potential challenge in big data area.

(3) *Dynamic sequence data:* In a wide range of applications, the processed data may be commonly dynamic but not static [53]. 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 [22, 30, 144], decremental mining [78], online progressive mining [60], and stream data processing [18, 53]. Most existing studies are designed without parallelization, research on parallelly mining sequences from dynamic sequence data has seldom been done so far. Therefore, there are still some challenges when parallelizing these mining methods, especially for parallelizing dynamic SPM algorithms.

(4) *Scalability:* In general, parallelizing SPM algorithms for dealing with big data is more complicated and difficult 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. Although a significant amount of developments, such as MG-FSM, LASH, and MLlib, have been reported, there is still much improvement for parallel implementation. There is still, however, 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 the one hand, policies that cover all user privacy concerns should be developed. On the other hand, privacy preserving data mining (PPDM) [147] and privacy preserving utility mining (PPUM) [39] are also needed. Up to now, many technologies and algorithms have been developed for PPDM [147] or PPUM [39], but few of them are related to PSPM. With the developments of SPM to analyze personal data, it is quite necessary and challenging to address the privacy preserving problem for PSPM.

7.2 Opportunities in PSPM

There are a variety of traditional, distributed, and PDM algorithms in numerous real-life applications. To meet the demand of large-scale and high-performance computing, the problem of parallel/DDM has received considerable attention over the past decade. There are many types of parallel and distributed systems, such as multi-core computing [29, 62, 100], grids [80, 83], peer-to-peer (P2P) systems [101], ad-hoc networks [127], cloud computing systems [47], and the MapReduce framework [27]. All these developments can provide theoretical and technical support for PSPM. 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 toward to extend PSPM in new applications, or in new ways for existing applications [33]. For example, the utility-driven mining [42] on large-scale sequence data is more practical but challenge.

(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 [47], MapReduce [27], and spark [135]. All these computing infrastructures provide new opportunities for the design of PDM, especially for PSPM algorithms. To some degree, all these developments can provide the necessary theoretical and technical support for PSPM.

(3) *Developments from hardware and software:* As mentioned before, a trend of parallel computing is GPU-based parallel techniques. Architecturally, a GPU is composed of hundreds of cores that can simultaneously handle thousands of threads/tasks [7, 16]. More and more supercomputers have the GPU component which is different from the traditional CPU. Therefore, we can expect an enhancement on PDM 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 is 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 [129], BIDE [120], MSPX [81], and ClaSP [48], as shown at Table 7) or summarization SPM (i.e., summarizing event sequences [114], SQUISH [15]) 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 [70] 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) [123], and Long Short-Term Memory (LSTM) [56]. Unlike the standard neural network, RNN allows us to operate over sequences of vectors [123]. 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 [56]. 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 SPM 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 PSPM in the big data era. How to design efficient and more

flexible PSPM approaches to support iterative and interactive mining is also an interesting issue.

8 CONCLUSION

Typically, SPM 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, PDM has emerged as an important research topic. However, fewer studies summarized the related development in PSPM in parallel computing environments, or summarized them as a taxonomy structure. In this article, 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 PSPM in several categories. The main contributions are that we investigate recent advances in PSPM and provide the status of the field in details including SPM, PFIM, and PSPM. Both basic algorithms and advanced algorithms for PSPM 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 PSPM for future research.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their detailed comments and constructive suggestions for this article.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *ACM SIGMOD Record* 22 (1993), 207–216.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the 7th International Conference on Data Engineering*. IEEE, 3–14.

[3] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*. 487–499.

[4] Oznur Kirmemis Alkan and Pinar Karagoz. 2015. CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction. *IEEE Transactions on Knowledge and Data Engineering* 27, 10 (2015), 2645–2657.

[5] George S. Almasi and Allan Gottlieb. 1989. Highly parallel computing. The Benjamin/Cummings Pub. Co., Inc. (3rd Edition).

[6] David C. Anastasiu, Jeremy Iverson, Shaden Smith, and George Karypis. 2014. Big data frequent pattern mining. In *Frequent Pattern Mining*. Springer, 225–259.

[7] Joshua A. Anderson, Chris D. Lorenz, and Alex Travesset. 2008. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics* 227, 10 (2008), 5342–5359.

[8] Daniele Apiletti, Elena Baralis, Tania Cerquitelli, Paolo Garza, Pietro Michiardi, and Fabio Pulvirenti. 2015. PaMPa-HD: A parallel MapReduce-based frequent pattern miner for high-dimensional data. In *Proceedings of the IEEE International Conference on Data Mining Workshop*. IEEE, 839–846.

[9] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential pattern mining using a bitmap representation. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 429–435.

[10] Roberto J. Bayardo Jr. 1998. Efficiently mining long patterns from databases. *ACM SIGMOD Record* 27, 2 (1998), 85–93.

[11] Kaustubh Beedkar, Klaus Berberich, Rainer Gemulla, and Iris Miliaraki. 2015. Closing the gap: Sequence mining at scale. *ACM Transactions on Database Systems* 40, 2 (2015), 8.

[12] Kaustubh Beedkar and Rainer Gemulla. 2015. LASH: Large-scale sequence mining with hierarchies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 491–503.

[13] Kaustubh Beedkar and Rainer Gemulla. 2016. DESQ: Frequent sequence mining with subsequence constraints. In *Proceedings of the IEEE 16th International Conference on Data Mining*. IEEE, 793–798.

[14] Pavel Berkhin. 2006. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*. Springer, 25–71.
[15] Apratim Bhattacharyya and Jilles Vreeken. 2017. Efficiently summarising event sequences with rich interleaving patterns. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 795–803.
[16] Sha'Kia Boggan and Daniel M. Pressel. 2007. *GPUs: An Emerging Platform for General-purpose Computation*. Technical Report. Army Research Lab Aberdeen Proving Ground Md Computational and Information Science Directorate, ARL-SR-154 (2007).
[17] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2008. BigTable: A distributed storage system for structured data. *ACM Transactions on Computer Systems* 26, 2 (2008), 4.
[18] Chun-Chieh Chen, Hong-Han Shuai, and Ming-Syan Chen. 2017. Distributed and scalable sequential pattern mining through stream processing. *Knowledge and Information Systems* 53, 2 (2017), 365–390.
[19] Chun-Chieh Chen, Chi-Yao Tseng, and Ming-Syan Chen. 2013. Highly scalable sequential pattern mining based on MapReduce model on the cloud. In *Proceedings of the IEEE International Congress on Big Data*. IEEE, 310–317.
[20] Jinlin Chen. 2010. An up-down directed acyclic graph approach for sequential pattern mining. *IEEE Transactions on Knowledge and Data Engineering* 22, 7 (2010), 913–928.
[21] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. 1996. Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering* 8, 6 (1996), 866–883.
[22] Hong Cheng, Xifeng Yan, and Jiawei Han. 2004. IncSpan: Incremental mining of sequential patterns in large database. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 527–532.
[23] David W. Cheung, Jiawei Han, Vincent T. Ng, Ada W. Fu, and Yongjian Fu. 1996. A fast distributed algorithm for mining association rules. In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*. IEEE, 31–42.
[24] Ding-Ying Chiu, Yi-Hung Wu, and Arbee L. P. Chen. 2004. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *Proceedings of the 20th International Conference on Data Engineering*. IEEE, 375–386.
[25] Shengnan Cong, Jiawei Han, Jay Hoeflinger, and David Padua. 2005. A sampling-based framework for parallel data mining. In *Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. ACM, 255–265.
[26] Shengnan Cong, Jiawei Han, and David Padua. 2005. Parallel mining of closed sequential patterns. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, 562–567.
[27] Jeffrey Dean and Sanjay Ghemawat. 2010. MapReduce: A flexible data processing tool. *Communications of the ACM* 53, 1 (2010), 72–77.
[28] Ayhan Demiriz. 2002. WebSPADE: A parallel sequence mining algorithm to analyze web log data. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 755–758.
[29] Romain Dolbeau, Stéphane Bihan, and François Bodin. 2007. HMPP: A hybrid multi-core parallel programming environment. In *Proceedings of the Workshop on General Purpose Processing on Graphics Processing Units*, Vol. 28.
[30] Maged El-Sayed, Carolina Ruiz, and Elke A. Rundensteiner. 2004. 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, 128–135.
[31] C. I. Ezeife, Yi Lu, and Yi Liu. 2005. PLWAP sequential mining: Open source code. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*. ACM, 26–35.
[32] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. 2016. The SPMF open-source data mining library version 2. In *Proceedings of the Joint European Conference on Machine Learning and knowledge Discovery in Databases*. Springer, 36–40.
[33] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage-Uday Kiran, Yun-Sing Koh, and Rincy Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.
[34] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Bay Vo, Tin-Truong Chi, Ji Zhang, and Hoai-Bac Le. 2017. A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7, 4 (2017), e1207.
[35] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S. Tseng. 2014. VMSP: Efficient vertical mining of maximal sequential patterns. In *Proceedings of the Canadian Conference on Artificial Intelligence*. Springer, 83–94.
[36] Philippe Fournier-Viger, Cheng-Wei Wu, and Vincent S. Tseng. 2013. Mining maximal sequential patterns without candidate maintenance. In *Proceedings of the International Conference on Advanced Data Mining and Applications*. Springer, 169–180.

[37] Jaroslav Fowkes and Charles Sutton. 2016. A subsequence interleaving model for sequential pattern mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 835–844.

[38] Fabio Fumarola, Pasqua Fabiana Lanotte, Michelangelo Ceci, and Donato Malerba. 2016. CloFAST: Closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems* 48, 2 (2016), 429–463.

[39] Wensheng Gan, Jerry Chun-Wei Lin, Han-Chieh Chao, Shyue-Liang Wang, and S. Yu Philip. 2018. Privacy preserving utility mining: A survey. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 2617–2626.

[40] Wensheng Gan, Jerry Chun-Wei Lin, Han-Chieh Chao, and Justin Zhan. 2017. Data mining in distributed environment: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7, 6 (2017), e1216.

[41] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Tzung-Pei Hong, and Hamido Fujita. 2018. A survey of incremental high-utility itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 2 (2018), e1242.

[42] Wensheng Gan, Jerry Chun-Wei Lin, Philippe Fournier-Viger, Han-Chieh Chao, Vincent S. Tseng, and Philip S. Yu. 2018. A survey of utility-oriented pattern mining. *arXiv:1805.10511*.

[43] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 1999. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases*, Vol. 99. 7–10.

[44] Jiaqi Ge and Yuni Xia. 2016. Distributed sequential pattern mining in large scale uncertain databases. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 17–29.

[45] Jiaqi Ge, Yuni Xia, and Jian Wang. 2015. Mining uncertain sequential patterns in iterative MapReduce. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 243–254.

[46] Liqiang Geng and Howard J. Hamilton. 2006. Interestingness measures for data mining: A survey. *Computing Surveys* 38, 3 (2006), 9.

[47] Lazaros Gkatzikis and Iordanis Koutsopoulos. 2013. Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems. *IEEE Wireless Communications* 20, 3 (2013), 24–32.

[48] Antonio Gomariz, Manuel Campos, Roque Marin, and Bart Goethals. 2013. ClaSP: An efficient algorithm for mining frequent closed sequences. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 50–61.

[49] Valerie Guralnik and George Karypis. 2004. Parallel tree-projection-based sequence mining algorithms. *Parallel Computing* 30, 4 (2004), 443–472.

[50] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2000. FreeSpan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 355–359.

[51] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering*. 215–224.

[52] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery* 8, 1 (2004), 53–87.

[53] Michael Bonnell Harries, Claude Sammut, and Kim Horn. 1998. Extracting hidden context. *Machine Learning* 32, 2 (1998), 101–126.

[54] Marwan Hassani and Thomas Seidl. 2011. Towards a mobile health context prediction: Sequential pattern mining in multiple streams. In *Proceedings of the 12th IEEE International Conference on Mobile Data Management*, Vol. 2. IEEE, 55–57.

[55] Scott Hauck and Andre DeHon. 2010. *Reconfigurable Computing: The Theory and Practice of FPGA-based Computation*, Vol. 1. Elsevier.

[56] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[57] Sungpack Hong, Tayo Oguntebi, and Kunle Olukotun. 2011. Efficient parallel graph exploration on multi-core CPU and GPU. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*. IEEE, 78–88.

[58] Tzung-pei Hong, Chan-Sheng Kuo, and Sheng-Chai Chi. 1999. Mining fuzzy sequential patterns from quantitative data. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3. IEEE, 962–966.

[59] Jen-Wei Huang, Su-Chen Lin, and Ming-Syan Chen. 2010. DPSP: Distributed progressive sequential pattern mining on the cloud. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 27–34.

[60] Jen-Wei Huang, Chi-Yao Tseng, Jian-Chih Ou, and Ming-Syan Chen. 2008. A general model for sequential pattern mining with a progressive database. *IEEE Transactions on Knowledge and Data Engineering* 20, 9 (2008), 1153–1167.

[61] Kevin A. Huck and Allen D. Malony. 2005. PerfExplorer: A performance data mining framework for large-scale parallel computing. In *Proceedings of the ACM/IEEE Conference on Supercomputing*. IEEE Computer Society, 41.

[62] Bao Huynh, Bay Vo, and Vaclav Snasel. 2017. An efficient method for mining frequent sequential patterns using multi-core processors. *Applied Intelligence* 46, 3 (2017), 703–716.
[63] Raymond Austin Jarvis and Edward A. Patrick. 1973. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on computers* 100, 11 (1973), 1025–1034.
[64] Yichuan Jiang and J. C. Jiang. 2014. Understanding social networks from a multiagent perspective. *IEEE Transactions on Parallel and Distributed Systems* 25, 10 (2014), 2743–2759.
[65] Mahesh V. Joshi, George Karypis, and Vipin Kumar. 2000. *Parallel Algorithms for Mining Sequential Associations: Issues and Challenges*. Technical Report. TR 00-002. Department of Computer Science, University of Minnesota, Minneapolis.
[66] Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Mahmoud Ali, Waleed Kamaleldin, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. 2014. Big data: Survey, technologies, opportunities, and challenges. *The Scientific World Journal* 2014, Article 712826 (2014), 18 pages.
[67] Chulyun Kim, Jong-Hwa Lim, Raymond T. Ng, and Kyuseok Shim. 2007. SQUIRE: Sequential pattern mining with quantities. *Journal of Systems and Software* 80, 10 (2007), 1726–1745.
[68] Alexandros Labrinidis and Hosagrahar V. Jagadish. 2012. Challenges and opportunities with big data. *Proceedings of the VLDB Endowment* 5, 12 (2012), 2032–2033.
[69] Guo-Cheng Lan, Tzung-Pei Hong, and Hong-Yu Lee. 2014. An efficient approach for finding weighted sequential patterns from sequence databases. *Applied Intelligence* 41, 2 (2014), 439–452.
[70] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.
[71] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. 2000. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review* 14, 6 (2000), 533–567.
[72] Feng Li, Beng Chin Ooi, M. Tamer Özsu, and Sai Wu. 2014. Distributed data management using MapReduce. *Computing Surveys* 46, 3 (2014), 31.
[73] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. 2008. PFP: Parallel FP-growth for query recommendation. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 107–114.
[74] Yen-hui Liang and Shiow-yang Wu. 2015. Sequence-growth: A scalable and effective frequent itemset mining algorithm for big data based on MapReduce framework. In *Proceedings of the IEEE International Congress on Big Data*. IEEE, 393–400.
[75] Vance Chiang-Chi Liao and Ming-Syan Chen. 2014. DFSP: A depth-first spelling algorithm for sequential pattern mining of biological sequences. *Knowledge and Information Systems* 38, 3 (2014), 623–639.
[76] Cheng-Hung Lin, Chen-Hsiung Liu, Lung-Sheng Chien, and Shih-Chieh Chang. 2013. Accelerating pattern matching using a novel parallel algorithm on GPUs. *IEEE Transactions on Computers* 62, 10 (2013), 1906–1916.
[77] Jerry Chun-Wei Lin, Wensheng Gan, Philippe Fournier-Viger, and Tzung-Pei Hong. 2015. RWFIM: Recent weighted-frequent itemsets mining. *Engineering Applications of Artificial Intelligence* 45 (2015), 18–32.
[78] Jerry Chun-Wei Lin, Wensheng Gan, and Tzung-Pei Hong. 2014. Efficiently maintaining the fast updated sequential pattern trees with sequence deletion. *IEEE Access* 2 (2014), 1374–1383.
[79] Jerry Chun-Wei Lin, Wensheng Gan, Tzung-Pei Hong, and Vincent S. Tseng. 2015. Efficient algorithms for mining up-to-date high-utility patterns. *Advanced Engineering Informatics* 29, 3 (2015), 648–661.
[80] Jiming Liu, Xiaolong Jin, and Yuanshi Wang. 2005. Agent-based load balancing on homogeneous minigrids: Macroscopic modeling and characterization. *IEEE Transactions on Parallel and Distributed Systems* 16, 7 (2005), 586–598.
[81] Congnan Luo and Soon M. Chung. 2005. Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 415–426.
[82] Congnan Luo and Soon M. Chung. 2012. Parallel mining of maximal sequential patterns using multiple samples. *The Journal of Supercomputing* 59, 2 (2012), 852–881.
[83] Ping Luo, Kevin Lü, Zhongzhi Shi, and Qing He. 2007. Distributed data mining in grid computing environments. *Future Generation Computer Systems* 23, 1 (2007), 84–91.
[84] Nizar R. Mabroukeh and Christie I. Ezeife. 2010. A taxonomy of sequential pattern mining algorithms. *Computing Surveys* 43, 1 (2010), 3.
[85] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. 1997. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* 1, 3 (1997), 259–289.
[86] Florent Masseglia, Pascal Poncelet, and Rosine Cicchetti. 2000. An efficient algorithm for web usage mining. *Networking and Information Systems Journal* 2, 5/6 (2000), 571–604.
[87] Suejb Memeti and Sabri Pllana. 2015. Accelerating DNA sequence analysis using Intel (R) Xeon Phi (TM). In *Proceedings of the IEEE Trustcom/BigDataSE/ISPA*, Vol. 3. IEEE, 222–227.
[88] Iris Miliaraki, Klaus Berberich, Rainer Gemulla, and Spyros Zoupanos. 2013. Mind the gap: Large-scale frequent sequence mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 797–808.

[89] Sandy Moens, Emin Aksehirli, and Bart Goethals. 2013. Frequent itemset mining for big data. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 111–118.
[90] Carl H. Mooney and John F. Roddick. 2013. Sequential pattern mining–Approaches and algorithms. *Computing Surveys* 45, 2 (2013), 19.
[91] Andreas Mueller. 1998. *Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison*. Technical Report. Retrieved on May 3, 2019 from http://hdl.handle.net/1903/437.
[92] Muhammad Muzammal and Rajeev Raman. 2015. Mining sequential patterns from probabilistic databases. *Knowledge and Information Systems* 44, 2 (2015), 325–358.
[93] Loan-T. T. Nguyen, Bay Vo, Tzung-Pei Hong, and Hoang-Chi Thanh. 2012. Classification based on association rules: A lattice-based approach. *Expert Systems with Applications* 39, 13 (2012), 11357–11366.
[94] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. 1995. Efficient parallel data mining for association rules. In *Proceedings of the 4th International Conference on Information and Knowledge Management*. ACM, 31–36.
[95] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, and et al. Pinto. 2004. Mining sequential patterns by pattern-growth: The PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering* 16, 11 (2004), 1424–1440.
[96] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. 2000. Mining access patterns efficiently from web logs. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 396–407.
[97] Shaojie Qiao, Tianrui Li, Jing Peng, and Jiangtao Qiu. 2010. Parallel sequential pattern mining of massive trajectory data. *International Journal of Computational Intelligence Systems* 3, 3 (2010), 343–356.
[98] Shaojie Qiao, Changjie Tang, Shucheng Dai, Mingfang Zhu, Jing Peng, Hongjun Li, and Yungchang Ku. 2008. PartSpan: Parallel sequence mining of trajectory patterns. In *Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, Vol. 5. IEEE, 363–367.
[99] J. Ross Quinlan. 2014. *C4.5: Programs for Machine Learning*. Elsevier.
[100] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis. 2007. Evaluating MapReduce for multi-core and multiprocessor systems. In *Proceedings of the IEEE 13th International Symposium on High Performance Computer Architecture*. IEEE, 13–24.
[101] Weixiong Rao, Lei Chen, Ada Wai-Chee Fu, and Guoren Wang. 2010. Optimal resource placement in structured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems* 21, 7 (2010), 1011–1026.
[102] Sanjay Rathee, Manohar Kaul, and Arti Kashyap. 2015. 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, 27–34.
[103] Matteo Riondato, Justin A. DeBrabant, Rodrigo Fonseca, and Eli Upfal. 2012. 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, 85–94.
[104] Guangchen Ruan, Hui Zhang, and Beth Plale. 2014. Parallel and quantitative sequential pattern mining for large-scale interval-based temporal data. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 32–39.
[105] Majed Sahli, Essam Mansour, and Panos Kalnis. 2013. Parallel motif extraction from very long sequences. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*. ACM, 549–558.
[106] Saber Salah, Reza Akbarinia, and Florent Masseglia. 2017. A highly scalable parallel algorithm for maximally informative k-itemset mining. *Knowledge and Information Systems* 50, 1 (2017), 1–26.
[107] Masakazu Seno and George Karypis. 2001. LPMiner: An algorithm for finding frequent itemsets using length-decreasing support constraint. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 505–512.
[108] Masakazu Seno and George Karypis. 2002. SLPMiner: An algorithm for finding frequent sequential patterns using length-decreasing support constraint. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 418–425.
[109] Takahiko Shintani and Masaru Kitsuregawa. 1998. Mining algorithms for sequential patterns in parallel: Hash based approach. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 283–294.
[110] Shijie Song, Huaping Hu, and Shiyao Jin. 2005. HVSM: A new sequential pattern mining algorithm using bitmap representation. In *Proceedings of the International Conference on Advanced Data Mining and Applications*. Springer, 455–463.
[111] Mauro Sousa, Marta Mattoso, and N. F. F. Ebecken. 1998. Data mining on parallel database systems. In *Proceedings of the International Conference on PDPTA: Special Session on Parallel Data Warehousing*.
[112] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the International Conference on Extending Database Technology*. Springer, 1–17.
[113] Andrew S. Tanenbaum and Maarten Van Steen. 2007. *Distributed Systems: Principles and Paradigms*. Prentice-Hall.
[114] Nikolaj Tatti and Jilles Vreeken. 2012. 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, 462–470.

[115] Minh-Thai Tran, Bac Le, and Bay Vo. 2015. Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently. *Engineering Applications of Artificial Intelligence* 38 (2015), 183–189.
[116] Roberto Trasarti, Francesco Bonchi, and Bart Goethals. 2008. Sequence mining automata: A new technique for mining frequent sequences under regular expressions. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 1061–1066.
[117] Chun-Wei Tsai, Chin-Feng Lai, Han-Chieh Chao, and Athanasios V. Vasilakos. 2015. Big data analytics: A survey. *Journal of Big Data* 2, 1 (2015), 21.
[118] Chun-Wei Tsai, Chin-Feng Lai, Ming-Chao Chiang, Laurence T. Yang, et al. 2014. Data mining for Internet of Things: A survey. *IEEE Communications Surveys and Tutorials* 16, 1 (2014), 77–97.
[119] Trang Van, Bay Vo, and Bac Le. 2018. Mining sequential patterns with itemset constraints. *Knowledge and Information Systems* 57, 2 (2018), 311–330.
[120] Jianyong Wang and Jiawei Han. 2004. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*. IEEE, 79–90.
[121] Ke Wang, Elaheh Sadredini, and Kevin Skadron. 2016. Sequential pattern mining with the micron automata processor. In *Proceedings of the ACM International Conference on Computing Frontiers*. ACM, 135–144.
[122] Ke Wang, Yabo Xu, and Jeffrey Xu Yu. 2004. Scalable sequential pattern mining for biological sequences. In *Proceedings of the 13th ACM international Conference on Information and Knowledge Management*. ACM, 178–187.
[123] Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1, 2 (1989), 270–280.
[124] Chih-Hung Wu, Chih-Chin Lai, and Yu-Chieh Lo. 2012. An empirical study on mining sequential patterns in a grid computing environment. *Expert Systems with Applications* 39, 5 (2012), 5748–5757.
[125] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. 2014. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering* 26, 1 (2014), 97–107.
[126] Yujie Xu, Wenyu Qu, Zhiyang Li, Geyong Min, Keqiu Li, and Zhaobin Liu. 2014. Efficient k-Means++ approximation with MapReduce. *IEEE Transactions on Parallel and Distributed Systems* 25, 12 (2014), 3135–3144.
[127] Yuan Xue, Baochun Li, and Klara Nahrstedt. 2006. Optimal resource allocation in wireless ad hoc networks: A price-based approach. *IEEE Transactions on Mobile Computing* 5, 4 (2006), 347–364.
[128] Xifeng Yan and Jiawei Han. 2002. gSpan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 721–724.
[129] Xifeng Yan, Jiawei Han, and Ramin Afshar. 2003. CloSpan: Mining closed sequential patterns in large datasets. In *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 166–177.
[130] Zhenglu Yang and Masaru Kitsuregawa. 2005. LAPIN-SPAM: An improved algorithm for mining sequential pattern. In *Proceedings of the 21st International Conference on Data Engineering Workshops*. IEEE, 1222–1222.
[131] Zhenglu Yang, Yitong Wang, and Masaru Kitsuregawa. 2007. LAPIN: Effective sequential pattern mining algorithms by last position induction for dense databases. In *Proceedings of the International Conference on Database Systems for Advanced Applications*. Springer, 1020–1023.
[132] Dongjin Yu, Wei Wu, Suhang Zheng, and Zhixiang Zhu. 2012. BIDE-based parallel mining of frequent closed sequences with MapReduce. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 177–186.
[133] Xiao Yu, Jin Liu, Xiao Liu, Chuanxiang Ma, and Bin Li. 2015. A MapReduce reinforced distributed sequential pattern mining algorithm. In *Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 183–197.
[134] Unil Yun. 2008. A new framework for detecting weighted sequential patterns in large sequence databases. *Knowledge-Based Systems* 21, 2 (2008), 110–122.
[135] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. 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, 2.
[136] Mohammed J. Zaki. 1999. Parallel and distributed association mining: A survey. *IEEE Concurrency* 7, 4 (1999), 14–25.
[137] Mohammed J. Zaki. 2000. Parallel and distributed data mining: An introduction. In *Proceedings of the Large-Scale Parallel Data Mining*. Springer, 1–23.
[138] Mohammed Javeed Zaki. 2000. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering* 12, 3 (2000), 372–390.
[139] Mohammed J. Zaki. 2001. Parallel sequence mining on shared-memory machines. *Journal of Parallel and Distributed Computing* 61, 3 (2001), 401–426.
[140] Mohammed J. Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning* 42, 1–2 (2001), 31–60.

[141] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, et al. 1997. New algorithms for fast discovery of association rules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, Vol. 97. 283–286.

[142] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, Paul Stolorz, and Ron Musick. 1997. Parallel algorithms for discovery of association rules. In *Proceedings of the Scalable High Performance Computing for Knowledge Discovery and Data Mining*. Springer, 5–35.

[143] Li Zeng, Ling Li, Lian Duan, Kevin Lu, Zhongzhi Shi, Maoguang Wang, Wenjuan Wu, and Ping Luo. 2012. Distributed data mining: A survey. *Information Technology and Management* 13, 4 (2012), 403–409.

[144] Binbin Zhang, Chun-Wei Lin, Wensheng Gan, and Tzung-Pei Hong. 2014. Maintaining the discovered sequential patterns for sequence insertion in dynamic databases. *Engineering Applications of Artificial Intelligence* 35 (2014), 131–142.

[145] Qiankun Zhao and Sourav S. Bhowmick. 2003. *Sequential Pattern Mining: A Survey*. Technical Report. 2003118. CAIS, Nayang Technological University, Singapore.

[146] Weizhong Zhao, Huifang Ma, and Qing He. 2009. Parallel K-means clustering based on MapReduce. In *Proceedings of the IEEE International Conference on Cloud Computing*. Springer, 674–679.

[147] Tianqing Zhu, Gang Li, Wanlei Zhou, and Philip S. Yu. 2017. Differentially private data publishing and analysis: A survey. *IEEE Transactions on Knowledge and Data Engineering* 29, 8 (2017), 1619–1638.

[148] Morteza Zihayat, Zane Zhenhua Hut, Aijun An, and Yonggang Hut. 2016. Distributed and parallel high utility sequential pattern mining. In *Proceedings of the IEEE International Conference on Big Data*. IEEE, 853–862.

Received April 2018; revised January 2019; accepted February 2019