Intelligent Data Analysis 17 (2013) 507–530
DOI 10.3233/IDA-130591
IOS Press

Review Article

Incremental mining of sequential patterns: Progress and challenges

Bhawna Mallick[a,*], Deepak Garg[a] and P.S. Grover[b]
[a]*Thapar University, Patiala, India*
[b]*Guru Tegh Bahadur Institute, New Delhi, India*

Abstract. Sequential pattern mining is a vital problem with broad applications. However, it is also challenging, as combinatorial high number of intermediate subsequences are generated that have to be critically examined. Most of the basic solutions are based on the assumption that the mining is performed on static database. But modern day databases are being continuously updated and are dynamic in nature. So, incremental mining of sequential patterns has become the norm.

This article investigates the need for incremental mining of sequential patterns. An analytical study, focusing on the characteristics, has been made for more than twenty incremental mining algorithms. Further, we have discussed the issues associated with each of them. We infer that the better approach is incremental mining on the progressive database. The three more relevant algorithms, based on this approach, are also studied in depth along with the other work done in this area. This would give scope for future research direction.

Keywords: Data mining, sequential patterns, sequence database, incremental mining, progressive database, progressive mining

1. Introduction

There has been an increase in our capabilities of both generating and collecting data with the progress and development of technology. The ultimate intent of this massive data collection is to have competitive benefits. This is achieved by determining unidentified patterns in data which further help in decision making. Therefore, developing approaches and tools to discover knowledge hidden in these databases is the need of the hour.

Data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation and knowledge representation are different steps of Knowledge Discovery in Databases (KDD). Of these, data mining is an extremely important step [34]. Mechanism of extracting information from data is shown in Fig. 1.

Data mining is the process to analyze the data with the objective of finding rules and patterns to categorize the data [37]. Classification, regression, summarization, clustering, association rule mining, sequential mining are some of the techniques used for data mining [36]. Of these, sequential pattern mining has emerged as an area of active research.

*Corresponding author: Bhawna Mallick, Head of the Department, Computer Science and Engineering, Galgotias College of Engineering and Technology, 1, Knowledge Park, Institutional Area, Phase II, Greater Noida, 201306, Uttar Pradesh, India. Tel.: +91 120 451 3800; Fax: +91 120 451 3880; E-mail: bhawna.mallick@gmail.com.

Table 1
A transaction database 'S'

Transaction identifier	Item sets
100	bread, butter, milk
200	bread, egg, milk
300	bread, milk, cheese
400	butter, cheese, sausage

Table 2
A sequence database 'D'

Sequence identifier	Customer purchase sequences
100	<bread(bread, butter, egg)(bread, egg) milk (egg, sausage)>
200	<(bread, milk) egg (butter, egg) (bread, cheese)>
300	<(cheese, sausage) (bread, butter) (milk, sausage) egg butter>
400	<cheese oats (bread, sausage) egg butter egg>

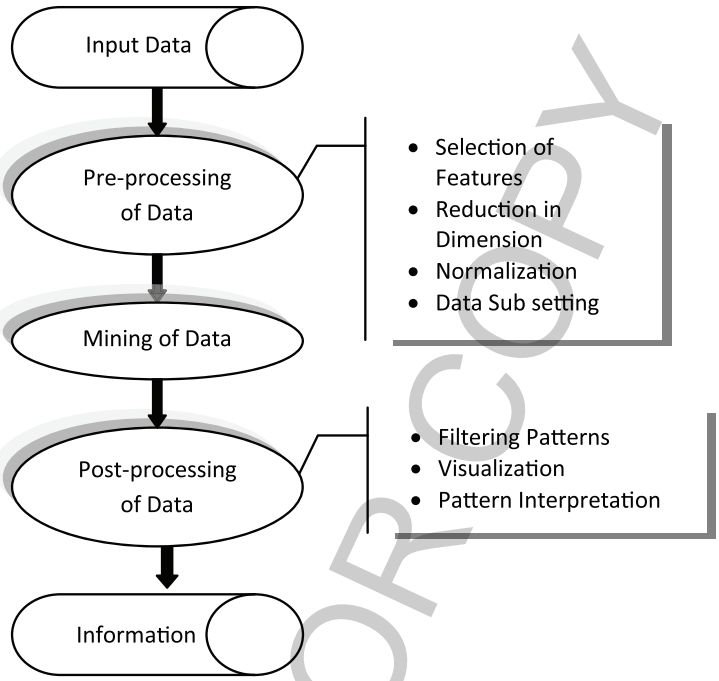

Fig. 1. Information extraction from the databases.

Sequential Pattern Mining [SPM] is the process that finds all frequent subsequences as patterns in a given sequence database [34]. A sequential pattern is a sequence having support greater than or equal to a minimum threshold, called the minimum support. The support of a subsequence is the percentage of data sequences containing the subsequence. The sequence database differs from the transaction database, as illustrated by the following example. This example, also examines the potential sequence patterns in a sequence database.

A simple transactional database includes a unique transaction identity number and a list of the items making up the transaction. Let our customer purchase transaction database be '***S***' given in Table 1. A transaction identifier *100* refers to a unique transaction with items *bread, butter, milk*.

On the contrary, a sequence database includes a sequence identity number and a sequence, where a sequence is an ordered list of events denoted as <e1 e2 e3 e4 ... en> where e1 occurs before e2, which occurs before e3 and so on. However the event by itself is an item set, that is, an unordered list of items denoted as $(x_1, x_2, \ldots, x_m)$ where x_j is an item. Let our customer purchase sequence database be '***D***' given in Table 2 with *100, 200, 300* and *400* be identifier for different sequences. A sequence 100 has ordered list of events <bread (bread, butter, egg) (bread, egg) milk (egg, sausage)> where purchase of

bread as an event e1 has occurred before purchase of another event e2 (bread, butter, egg). The event e2 is an item set consisting of unordered list of items – bread, butter and egg.

We can conclude with defining, $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals called items. An item set is a non-empty set of items. A sequence ***s*** is a set of item sets ordered according to their time stamp. It is denoted by $[s_1, s_2, \ldots, s_n]$, where $s_j, j \in 1, \ldots, n$, is an item set. A k-sequence is a sequence of k items (or of length k). A sequence $[s_1, s_2, \ldots, s_n]$ is a sub-sequence of another sequence $[s'_1, s'_2, \ldots, s'_m]$, if there exist integers $i_1 < i_2 < \ldots i_j \ldots < i_n$; such that $s_1 \subseteq s'_{i1}, s_2 \subseteq s'_{i2}, \ldots, s_n \subseteq s'_{in}$.

Sequential pattern is a sequence of item sets that frequently occur in a specific order. All items in the same item set are supposed to have the same transaction time value or within a time gap [34,37]. Each sequence corresponds to a temporally ordered list of events, where each event is a collection of items (item set) occurring simultaneously. The problem of SPM is to find all frequent subsequences whose occurrence frequency is no less than the user defined minimum threshold ***minSupp***. For our example, <bread (butter, egg) milk egg> is a subsequence of <bread (bread, butter, egg) (bread, egg) milk (egg, sausage)>. Given a minimum support threshold, minSupp = 2, <(bread, butter) egg> is a sequential pattern for our example in Table 2.

SPM generally faces challenges due to large search space and lack of prior information for the number of items in an item set and/or the number of item sets in a pattern. The pattern could be formed by any possible permutation and combination of both single items and item sets in the database.

Sequential pattern mining finds application in various areas like telecommunication industry to study customer call pattern, financial institutions like banks for analysis of financial data, retail industry to analyze the sales data, stock market analysis and prediction, biological data analysis as in DNA sequence analysis, scientific applications as weather prediction, intrusion detection in networks, world wide web as to study web traversal sequences.

1.1. Motivation for the study

SPM and the different algorithms for its implementation have been studied by number of researchers [16,17,34,36]. This paper contributes to the research because:

1. It discusses the major drawback of the basic algorithms that are used for SPM.
2. It is the first attempt that studies the various approaches and issues of incremental SPM.
3. It also presents the need of incremental mining on progressive databases to have more relevant sequential patterns.

The goal of the paper is to answer questions like:

- What do we mean by incremental SPM?
- What is the need of incremental SPM?
- What are the different approaches used by incremental SPM algorithms?
- What are the different issues with the incremental SPM algorithms?
- How we can overcome the drawbacks of these algorithms?

1.2. Organization of the paper

The remainder of the paper is organized as follows. Section 2 presents in brief the basic algorithms used for the SPM and the problem associated with them. The need for incremental mining of sequential patterns is discussed in Section 3. The different approaches of incremental mining algorithms are studied in Section 4. In Section 5 the comparative analysis is made for these algorithms on the basis

of their characteristics. The issues of incremental mining algorithms for SPM are discussed in detail in Section 6. The different approaches that can resolve the limitation of incremental mining are discussed in Section 7. Section 8 gives the application of these algorithms with the help of real time examples. We have concluded our study with the scope for future research in Section 9.

2. Basic approaches for SPM

There is plenty of literature available on different approaches used by SPM algorithms [2,8,12,15–18, 21,27,34,36,38,45–47,49]. We briefly present some of the salient approaches for the completeness of the article.

AprioriSome, AprioriAll and Dynamicsome algorithms were proposed by Agrawal and Srikant [2, 36,37] based on apriori approach. The approach uses the prior knowledge of frequent sequences. It makes use of iterative approach for candidate generation. It starts with the generation of frequent 1 sequences and then generates frequent $(k + 1)$ sequences from the set of frequent k sequences (called as candidate). However to decide whether a particular sequence is frequent, the entire database is scanned. The GSP (Generalized Sequential Pattern) [34] algorithm is an extension of the apriori model (candidate generation approach) and is based on sliding window principle. The items are considered in the same transaction if the distance between the maximal transaction time and the minimal transaction time of the items is no bigger than the sliding window.

SPADE (Sequential Pattern, Discovery using Equivalent classes) algorithm [27] is based on apriori vertical formatting method. The sequential database is converted into a vertical id-list database format and each id is associated with corresponding items and the time stamp. This algorithm aims to find frequent sequences using efficient lattice search techniques and simple joins. All the sequences are discovered with only three passes over the database. The first scan finds the frequent items, the second scan search for the frequent sequences of length 2 and the last one associate to frequent sequences of length 2, a table of the corresponding sequences id and item sets id in the database. The method forms the enumeration of the candidate sequences, based on their common prefixes. As AprioriAll, GSP and other algorithms depend little on main memory so SPADE outperforms these algorithms.

Later, projection based pattern growth approach was developed that avoids the candidate generation, and search is applied only on a restricted part of the database. FreeSpan [17] is the first algorithm proposed that considers the projection method for SPM. This work has been continued with Prefix Span algorithm [21]. The algorithm starts from the frequent items of the database, and generates projected databases with the remaining data-sequences. The projected databases thus contain suffixes of the data-sequences from the original database, which are grouped by prefixes. The process is recursively repeated until no frequent item is found in the projected database. By using projection, the database PrefixSpan scans every time is much smaller than the original database. Unlike GSP and AprioriAll, PrefixSpan can handle very long sequential pattern more efficiently. This method is also called level-by-level projection.

PSP algorithm [8,17] has implemented pseudo projection technique, which do not construct projection database physically. Each postfix is represented by a pair of pointer and offset value in the main memory. However, it can be used only when the projected database and its associated pseudo-projection processing structure fits in the main memory. SPAM (Sequential PAttern Mining) [12] is another method that represents the database in the main memory. It generates a vertical bitmap representation of the database for both candidate representation and support counting.

SPaRSe (Sequential PAttern mining with Restricted SEarch) [34,38] is an algorithm that makes apriori-based method as efficient as pattern-growth method under specific conditions. It uses both the

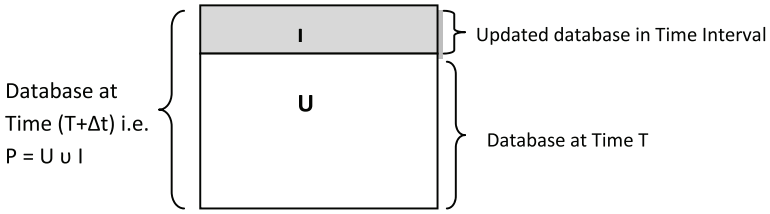

Fig. 2. Changes in database due to dynamic transactions.

candidate generation and projected database concept to achieve higher efficiency for high pattern density conditions. The algorithm maintains a list of supporting sequences for each candidate. It then verifies the existence of support only in the subset of sequences that are super-sequences of generated candidates [17]. MEMISP (MEMory Indexing for Sequential Pattern mining) [16,34] algorithm requires one pass over the database and at most two passes for very large database. It avoids candidate generation and intermediate database projection by implementing techniques of recursive searching and indexing strategy, to generate the sequential patterns from the data sequences that are stored in memory.

In general, SPM algorithms either result in enormous or very few patterns based on the user specified support threshold. Almost no or very few sequential patterns are given in case of large support threshold value. On the contrary, the mining algorithms result with enormous patterns for a smaller support threshold value. These methods hence make a tradeoff between efficiency and effectiveness. Further, we may list the limitations of these basic approaches as below:

a) There are often a huge number of sequential patterns in a large database. The complete set of sequential patterns generated makes the mining result ineffective and difficult to use.
b) They do not focus only on those sequential patterns that are of interest to users, and hence are computationally expensive.
c) They assume that the data is centralized, memory-resident and static. So they waste computational and input/output (I/O) resources for dynamic data and impose excessive communication overhead for distributed data.

3. Incremental Sequential Pattern Mining (ISPM)

In practice, databases are dynamic and are appended with new data sequences, by either existing or new customers. Some of the existing patterns may become invalid as their minimum support become insufficient with respect to the currently updated database. Some new patterns may even get created due to the increase in their support value. The incremental sequential pattern mining is required to handle this situation where the result mined from the old database is not valid on the updated database and it is inefficient to mine the updated databases from scratch [9,11].

We can define the Incremental Sequential Patterns Mining (ISPM) as the process to compute the set of sequential patterns from the updated database by using the two operations stated as;

a) Discover all sequential patterns which were not frequent in the original database but become frequent with the increment.
b) Examine all transactions in the original database that can be extended to become frequent.

For the purpose of representation, let us consider an original database 'U' and an incremental database 'I' where new transactions or new customers are added to 'U' (Fig. 2). The minimum support threshold

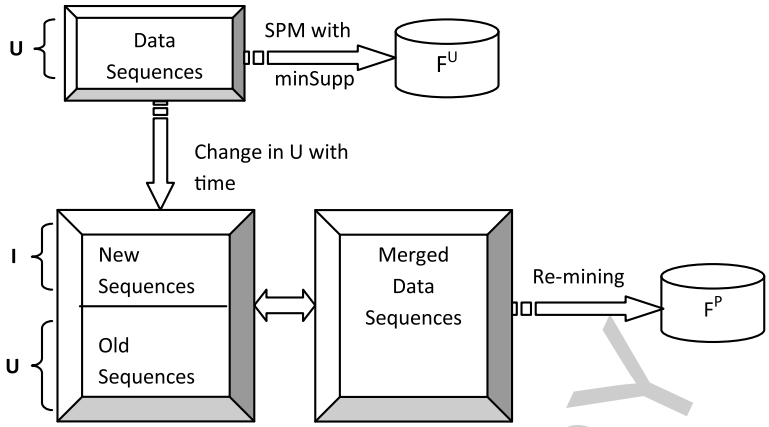

Fig. 3. Process flow to obtain F^P by re-mining the database using SPM algorithms.

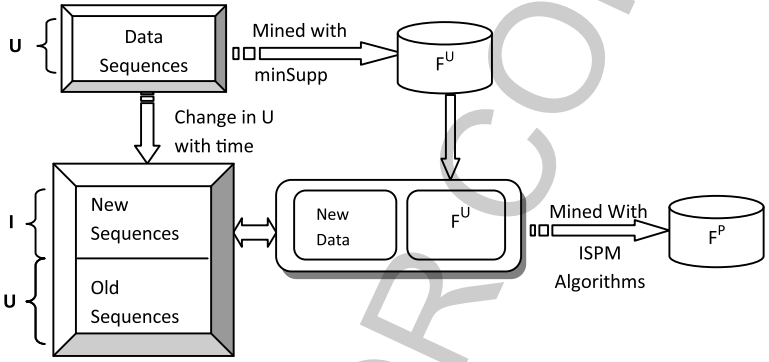

Fig. 4. Process flow to obtain F^P by ISPM algorithms.

is represented as *minSupp*. $P = U \cup I$ is the updated database containing all sequences from U and I. If F^U *is* the set of frequent sequences in U, then the problem of ISPM is to find frequent sequence patterns in P, denoted as F^P, with respect to the same minimum support. Unlike the conventional static mining algorithms, the incremental approach avoids re-running of mining algorithms from scratch when the database is updated (Fig. 3). ISPM algorithms reuse previously mined information and combine this information with the fresh data to efficiently compute the new set of frequent sequences [9] as depicted in Fig. 4.

Therefore, developing efficient approaches for ISPM has received a great deal of attention for ensuring scalability and facilitating knowledge discovery when data is dynamic and distributed.

4. Approaches for ISPM algorithms

In recent times, ISPM algorithms have gained immense importance in real life applications. A concise review of the research work related to the ISPM is presented here. Based on the typical naïve approach each of these algorithms implement and the characteristics they possess, we have classified them in four categories (Fig. 5) as given in the following sub-sections.

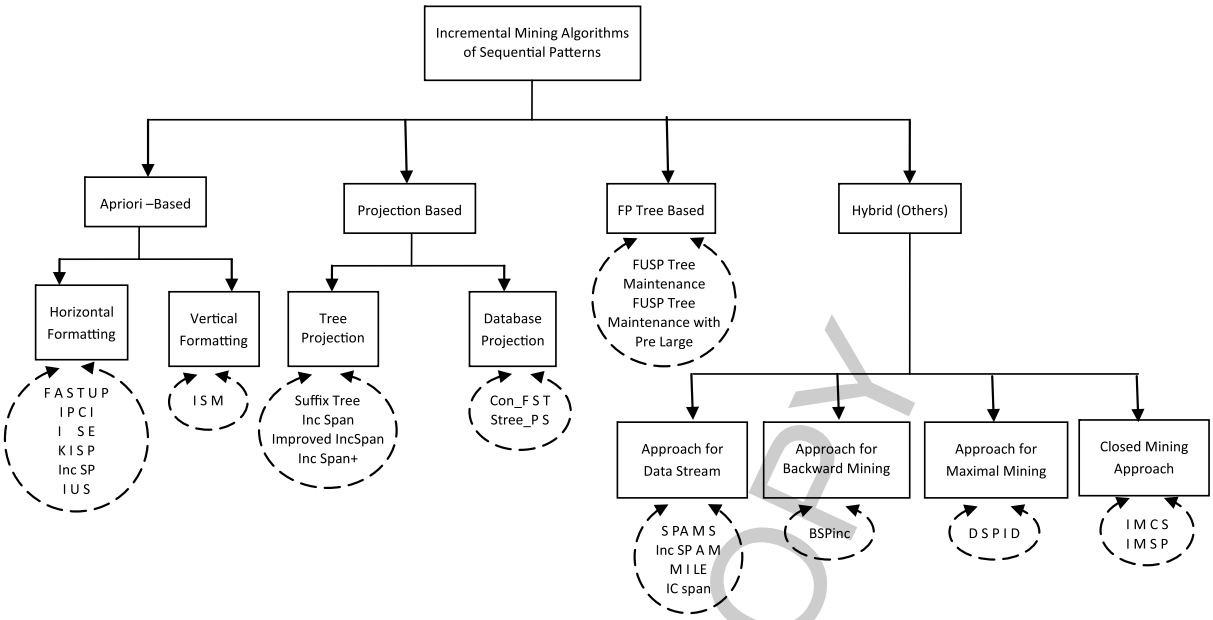

Fig. 5. Taxonomy of ISPM algorithms.

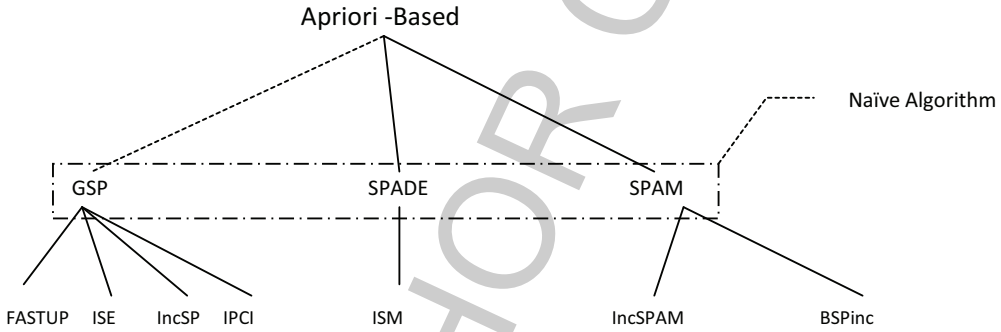

Fig. 6. ISPM algorithms that use Apriori as naïve approach.

4.1. Apriori-based approach

The ISPM algorithms discussed in this section are those that have used Apriori-based (candidate generate-and-test) approach. Figure 6 gives the diagrammatic representation of these set of ISPM algorithms.

FastUP approach [28] is one among the initial work done in the field of the ISPM. It is based on GSP approach with improvements performed on candidate generation and support counting. The algorithm uses the pruning method before candidate generation and validation. This pruning approach uses the previous mining results to get information about the sequence thresholds. Hence it can avoid generating some sequence based on their support. It further applies techniques of fast filtering during support counting and successive candidate sequence reduction. This improves the overall space and time efficiency of the algorithm over conventional SPM algorithm.

Then, an algorithm with an approach similar to SPADE was proposed called ISM (Incremental Se-

quence Mining) [40], which maintains sequence lattice of an old database. This lattice includes all the sequences that are frequent or are in the negative border. The negative border concept helps algorithm to maintain the updating of the database. ISM performs pruning of lattice and generates negative border sequences in one database scan initially. In next step, the new frequent sequences are generated using the SPADE approach. As the number of sequences in the negative border can be very huge, this approach could be expensive in terms of memory and time. However, it is unlikely to have sequences in the negative border to become frequent in the updated database if they have low support.

IUS (Incrementally Updating Sequences) [35] is an algorithm that also minimizes the computing cost by reusing the frequent and negative border sequences in the original database. A new threshold for negative border sequences called the minimum support of negative border sequences; min_nbd_Supp is defined by the algorithm to control the memory and time consumed otherwise by these sequences [as observed in ISM algorithm]. Hence the frequent sequences whose support is less than min_Supp and greater than min_nbd_Supp are treated as negative border sequences. IUS generates candidates from original database by extending both the prefix and suffix of frequent sequences.

Another well recognized ISPM algorithm ISE (Incremental Sequence Extraction) [9] minimizes the computational effort by using the old frequent sequences in an iterative manner and then applying optimization techniques. ISE algorithm though, significantly outperforms the naïve approach, GSP algorithm but suffers from major limitations. The problem of this algorithm is (a) the candidate set can be very huge making the test-phase very slow; and (b) its level-wise working manner requires multiple scans of the whole database. This is very costly, especially when the sequences are long.

IncSP (Incremental Sequential Pattern Update) [29] algorithm is based on combined approaches of effective implicit merging, early candidate pruning and efficient separate counting. Implicit merging of updated sequences with previous useful sequences is done. Candidate pruning is performed by checking counts in the increment database to generate fewer but better candidates. However, support counting of promising candidates over original database gives new updated patterns. IncSP not only updates the support of existing patterns but also find out the new patterns from the updated database in continuous manner. IncSP was found to be fast and scalable as compared to GSP and ISM in most cases.

The concept of partitioning the search space using theory of rough sets is used by the ISPM algorithm, IPCI [13]. The knowledge of the minimal generators (one that do not have any sub patterns), frequent closed patterns (the one with no super pattern with the same support) and maximal patterns (sequence set with the highest cardinality) is used. All these types of patterns with their occurrence frequency are utilized to reduce the space of frequent sequential patterns in the updated database. IPCI looks for patterns in only the specified equivalence classes and require no buffering of patterns. This gives better time and space efficiency. The database scans are limited to one or two with no huge candidate generation.

The SPM process is typically iterative and interactive. As discussed in Section 2, large or very few patterns could be mined from the database depending on the minimum support (minSupp) value. So generally the user tries various minSupp values until the result is satisfactory. Most ISPM approaches are not designed to deal with repeated mining so each minSupp value invokes a re-mining from scratch. A simple approach, called KISP (Knowledge base assisted Incremental Sequential Pattern) [30] was proposed to improve the efficiency of sequential pattern discovery with changing supports. KISP utilizes the information obtained from prior mining performed using an assumed least minSupp. It then generates a knowledge base (abbreviated *KB*) for further queries about sequential patterns of various minSupp. In cases when the results cannot be directly obtained from the KB, KISP performs fast sequence discovery by eliminating the candidates existing in KB, before counting support. The algorithm can accepts any

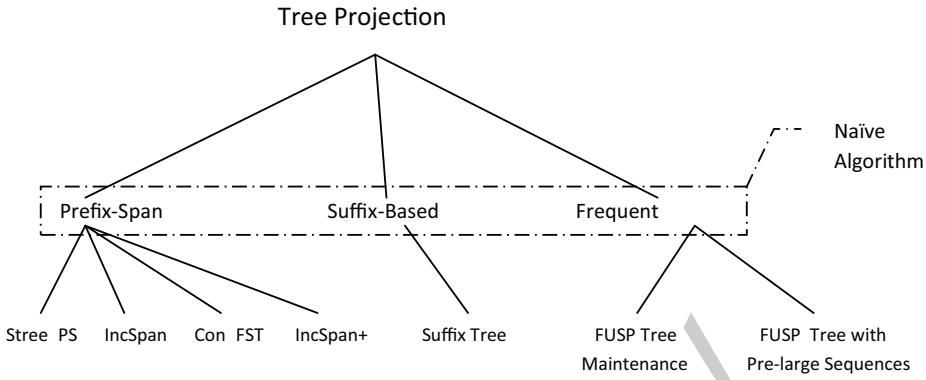

Fig. 7. ISPM algorithms that use Tree Projection as naïve approach.

minSupp value and has no difficulty in mining huge databases even with a small main memory. Two optimizations are done to speed up KISP. The direct generation of new candidates that eliminates candidate searching in KB and the concurrent support counting that reduces database scanning.

4.2. Projection based pattern – growth approach

ISPM algorithms that are based on tree projection approach (Fig. 7) are discussed in this section. These algorithms do not generate candidate set followed by testing phase and hence avoid rescanning of the database, as is the case with the apriori-based methods.

Suffix Tree algorithm [24], a very preliminary concept that use projection approach was first introduced for ISPM. It maintains suffix tree to store the frequent sequences of the database. The algorithm suggests that for every addition of the data sequence in the database, the tree could be modified by inserting a node appropriately. However, the focus was on multiple sequences of item sets, instead of a single long sequence of items.

Then, a popular algorithm IncSpan [11], based on PrefixSpan approach was proposed that has interesting properties. When mining the original database, IncSpan buffers the frequent sequences, as well as the sequences that are *semi-frequent*, which are likely to become frequent in the new version. Later, when mining the new version, only sequences with support over a certain threshold are likely to become frequent, out of un-buffered sequences. The algorithm has improved performance due to effective implicit merging, early candidate pruning, and efficient separate counting. However, it was found that IncSpan is not general enough to handle real life database evolution and its performance is far from optimal.

IncSpan algorithm fails to mine the complete set of sequential patterns from an updated database, as discovered by Nguyen et al. [39]. They clarified the weakness of the algorithm, by proving the incorrectness of the basic properties in the IncSpan algorithm. Also, they rectified the observed shortcomings by giving their solution as Improved IncSpan.

Another better solution to avoid the drawback of IncSpan algorithm was given by Chen et al. [48]. IncSpan+, an ISPM algorithm constructs a prefix tree to represent the sequential patterns. Then it continuously scans the incremental item set in the updated database and maintains the tree structure accordingly. To improvise the performance and eliminate the search space, the techniques of width pruning and depth pruning are used by this algorithm.

In general, the ISPM algorithms that are based on tree projection, maintain structure to store frequent sequential patterns with their minimum support threshold. With every update in the original database,

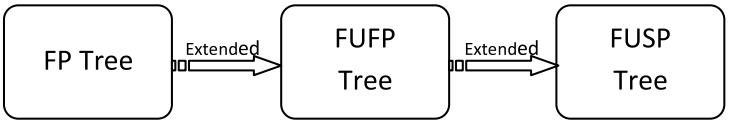

Fig. 8. Evolution of Frequent Pattern Tree for maintaining sequential patterns.

these algorithms re-construct the storage structure. Liu et al. [19] proposed an algorithm Stree_PS (Sequence tree_PrefixSpan) to construct and maintain a storage structure called sequence tree. This tree is similar to prefix tree which stores all sequences in the original database i.e. all frequent and infrequent sequences along with their support. Hence the path from the root node to leaf node represents a sequence in the database. Stree_PS adds the incremental sequences to the sequence tree and dynamically maintain its structure. It uses depth first search strategy to traverse the sequence tree and find all sequential patterns in the updated database. This avoids re-constructing of projected database with every updating of original database. Hence, the algorithm has better time and space efficiency for ISPM.

Further enhancement in this direction was made by introducing a new storage structure called frequent sequence (FS) tree [20]. The data storage structure used by most of the ISPM algorithms cannot cope up with the change in the minimum support threshold given by the user. In such cases, the algorithms need to mine the database again. The FS tree stores all the sequential patterns that satisfy the minimum support threshold along with their support. An algorithm called Con_FST (Construction algorithm for Frequency Sequence Tree) can find all the sequential patterns with change in support, without re-mining the database, using this tree. The path from the root node to any leaf node represent sequential pattern in the database. The root node of the FST stores the support threshold of the frequent sequence tree. To enhance the performance of the algorithm, the pruning strategy could be further applied. However, for high support threshold, the number of sequential pattern is limited and the storage requirement of projected database is less than that of the sequence tree.

4.3. Frequent pattern tree based approach

Another set of ISPM algorithms, are those that use frequent pattern (FP) tree (different from frequent sequence tree) [5] as storage structure. A FP tree was extended to fast updated frequent pattern (FUFP) tree which further was upgraded to fast updated sequential pattern (FUSP) tree as storage structure (Fig. 8).

The proposed FUSP tree structure extends the concept of FUFP – tree and IncSpan tree to store only the frequent sequential patterns. This makes the FUSP tree compact and complete to avoid the rescanning of the original database. An incremental FUSP – tree maintenance algorithm [5] was developed to handle and maintain the FUSP – tree for recursive ISPM. This can handle the new transactions from old customers and new customers. The FUSP – tree is build initially with the original database. When new transactions are added, the algorithm processes them to maintain the FUSP tree. The entire FUSP tree is re-constructed in a batch manner when a sufficiently large number of transactions have been inserted. As the frequency of sequences is not maintained by FUSP-tree, the tree complexity is not optimal.

Hong et al. [44] have proposed an algorithm, called Incrementally Fast Updated Sequential Pattern tree with pre-large sequences. It modifies the FUFP tree maintenance algorithm, based on pre-large concept [43], to reduce the rescanning of the original database, till a sufficiently large number of sequences are added. This modified FUFP tree structure is extended to FUSP-tree to store the frequent sequential patterns. The FUSP-tree is initially built from the original database. The algorithm divides the newly added customer sequences into three parts as large, pre-large or small sequences of the original database

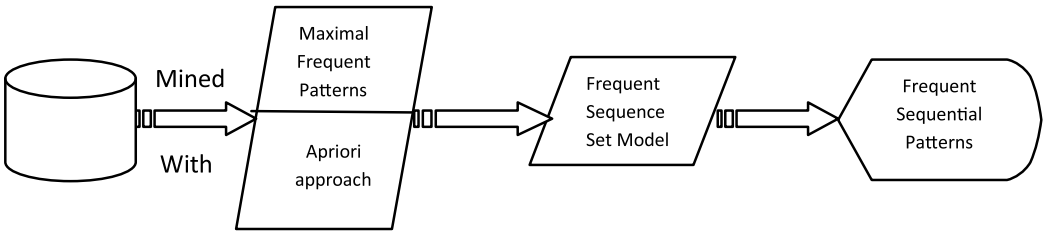

Fig. 9. Combined approaches used by DSPID algorithm in its implementation.

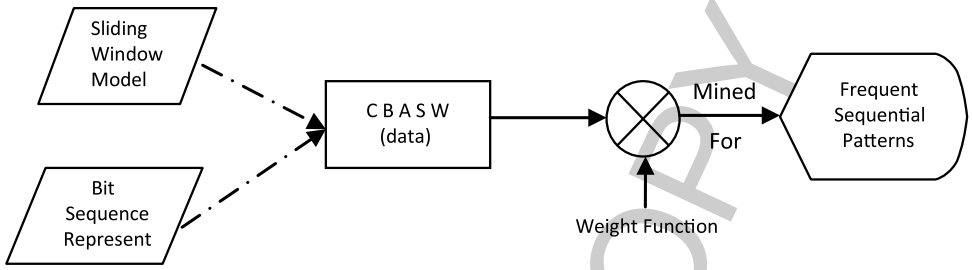

Fig. 10. Combined approaches used by IncSPAM algorithm in its implementation.

Fig. 11. Combined approaches used by IMCS algorithm in its implementation.

and processes them accordingly. Then these sequences are sorted in descending order of their frequency. Finally they are scanned again to reconstruct the FUSP tree according to their sequence order. The algorithm also makes use of index table and header table to store the index and frequency of these sequences.

4.4. Hybrid/other approaches

This section studies those ISPM algorithms that are based on combination of typical approaches (Figs 9–13) and hence have interesting properties.

The algorithm DSPID (Discover Sequential Patterns in Incremental Database) [33] uses apriori and maximal sequence pattern approach (Fig. 9) with some variations. It utilizes the information obtained from previous mining results unlike apriori-based approaches. It performs mining of maximal sequence patterns (compact representation of frequent sequential patterns) that reduce the number of patterns, without losing any information. DSPID then builds a data model called Frequent Sequence Set (FSS), to represent original sequence database along with frequency. The algorithm compares the new sequence appended in the database with the existing sequences in the FSS. Every new sequence is processed as per the relationship; it has with the original sequences as defined by the algorithm. It saves a lot of memory because no candidate sets are generated during mining process.

The algorithm IncSPAM [6] is typically used for SPM over a stream of item set sequences. It utilizes the sliding window model over data stream, that is, transaction sensitive. The model receives the transactions from the data stream and uses an effective bit sequence representation of item set. This

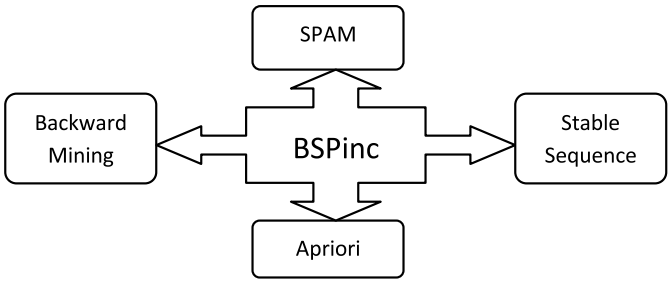

Fig. 12. Combined approaches used by BSPinc algorithm in its implementation.

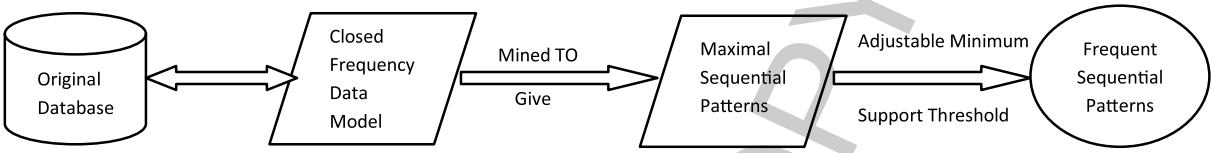

Fig. 13. Combined approaches used by IMSP algorithm in its implementation.

bit vector called Customer Bit Vector Array with Sliding Window (CBASW) can analyze each transaction effectively. The algorithm then makes use of weight function, to evaluate the importance of a sequence (Fig. 10) and also ensure the correctness of the sliding window model. IncSPAM is a single pass algorithm with efficient execution time and memory requirement.

Another algorithm, MILE (MIning in muLtiple strEams) [10] was proposed for mining sequential patterns in multiple data streams. The performance of the algorithm is improvised, by having the prior information regarding the data distribution in data streams. This avoids re-scanning of the database. MILE though based on PrefixSpan, append suffix with updated sequences to enhance the speed of discovering the new frequent sequential patterns. It emphasizes on generating sequential patterns from periodical data and uses a hash method to reduce time requirement. So the sequential patterns generated are not complete and accurate.

Chang et al. [25] have examined the problem of mining the closed sequential patterns incrementally. The idea is to mine patterns which have no super-patterns with the same support in the database [15,18]. This reduces the redundant patterns generation and subsequently the storage requirement, as compared to complete sequential pattern mining. The efficiency of mining result is substantial if the minimum support value is low. A compact structure CSTree (closed sequence tree) was designed to keep the closed sequential patterns of the database. They proposed an efficient algorithm IMCS, (Incremental Maintenance of Closed Sequential Pattern) that maintains the CSTree for the sequence database, that is updated incrementally. The nice properties of CSTree are used to find the obsolete sequences and to change the state of other sequences (represented as nodes), when the database is updated (Fig. 11).

ICspan (Incremental mining of closed sequential patterns in multiple data streams) [41] algorithm has also implemented the concept of closed sequential pattern mining to resolve the drawback of MILE algorithm. MILE algorithm with every new stream data input, mines only the new data rather than integrating new mining results with old ones. ICspan algorithm, hence proposed the use of sliding window model that sample the data from the input stream into units. It integrates the mining result obtained earlier with new result by implementing closed SPM in an incremental manner. So, the memory and time requirement of the algorithm is better than that of MILE.

BSPinc (Backward SPAM for incremental mining) algorithm is based on backward mining concept, for efficient incremental discovery of sequential patterns [31]. Candidates are generated as $(k + 1)$ sequences from k patterns, within the shrinking space of the projected database in the backward manner. The bitmapped representation is used for sequences. The patterns whose support count does not change in the updated database are mined with forward mining approach followed by apriori based method. BSPinc makes use of stable sequence property (Fig. 12) for candidate pruning and hence substantially improves the most time consuming step of support counting.

There is another algorithm IMSP that performs only one database scan to find the frequent sequential patterns and generates no candidate sets. This is done by transforming the original sequence database into a closed frequency data model. It mines frequent sequences in form of maximal sequential pattern instead of mining the full set of frequent sequences. IMSP has compact searching space compared to other maximal pattern mining algorithm and adjustable minimum support threshold (Fig. 13). The approach requires less memory space.

SPAMS (Sequential Patterns Automaton for Mining Streams), is an online algorithm proposed [26] for incremental sequential patterns mining in data streams. This algorithm uses SPA (Sequential Pattern Automaton), an automaton-based structure to maintain frequent sequential patterns. The automaton indexes all frequent sequential patterns from a data stream. The number of these patterns is smaller than the set of otherwise generated frequent sequential patterns by two or more orders of magnitude. This helps to overcome the problem of combinatorial high number of sequential patterns, usually in case of other algorithms for data streams. The proposed algorithm, SPAMS, is designed by considering the characteristics of data streams. The results are based on the users' specified threshold. Experimental studies performed, showed the relevance of the SPA data structure and the efficiency of the SPAMS algorithm, on various datasets. However, this contribution is an initiative of using an automaton as a data structure for mining, based on some approximation.

The pre-large concept [43] is generally used to postpone original small sequences, becoming large directly and vice versa, when new transactions are added. This basic idea was used for ISPM very recently. A lower support threshold and an upper support threshold are defined for the pre-large sequences. The upper support threshold is same as the minimum support threshold of conventional mining algorithms. The support ratio of a sequence must be larger than the upper support threshold to be considered large. A sequence with a support ratio below the lower threshold is treated as a small sequence. Hence, pre-large sequence acts like buffer to reduce the movement from large to small and small to large in incremental mining. Therefore when few new transactions are added in the database, the original small sequences at most become pre-large and never large. This reduces the rescanning of the original database. Both the threshold – upper and lower along with the size of the database define the safety bound for new transactions.

The theoretical basis of the approaches used by various authors to investigate ISPM algorithms have been summarized in Table 3 for the quick reference.

5. Analysis of ISPM algorithms

The study done on these ISPM algorithms helps us to draw important inferences and specify characteristic features of each one of these. This has been given in Table 4.

We can see from this table, IncSpan+, BSPinc, SPAMS, IPCI, ISM are the algorithms whose performance depends on size of frequent patterns. They become more useful in application where the size

Table 3
Concepts used by ISPM algorithms in their implementation

Year	ISPM algorithm	Theoretical Basis
1996	Suffix tree [Wang et al.]	Use a suffix tree technique to store the frequent sequential patterns.
1998	FASTUP [Lin et al.]	Use a candidate generation and support counting approach with candidate pruning.
1999	ISM [Parthasarthy et al.]	SPADE based approach that maintains sequence lattice of frequent sequences and sequences in negative border.
2002	IUS [Zheng et al.]	Based on negative border approach and uses a minimum support of negative border sequences threshold along with min_Supp.
2003	ISE [Masseglia et al.]	Use a candidate-generate and test approach of Apriori based algorithms.
2003	KISP [Lin et al.]	Interactive mining to obtain results for different values of minimum support. It maintains knowledge âĂŞ base for this purpose.
2004	IncSpan [Cheng et al.]	PrefixSpan based approach and uses implicit merging, early candidate pruning and efficient separate counting.
2004	IncSP [Lin et al.]	Implicit merging, separate counting over appended sequences and early candidate pruning for incrementally updated patterns.
2005	Improved IncSpan [Nguyen et al.]	Rectified the incorrectness of the basic properties of IncSpan.
2005	MILE [Chen et al.]	Based on prefixSpan but also append suffix to discover patterns in multiple data streams.
2006	IncSPAM [Ho et al.]	Uses the sliding window model over data streams and bit vector representation for the customer sequential item sets.
2007	IncSpan+ [Chen et al.]	Maintains prefix tree to represent sequential patterns and implement width and depth pruning to eliminate search space.
2007	DSPID[Lin et al.]	Based on maximal sequential pattern mining and uses a breadth first technique to extract the frequent maximal patterns.
2007	IMCS [Chang et al.]	Mine closed sequential patterns incrementally using CSTree as the database is updated in an incremental manner.
2008	FUSP tree maintenance [Lin]	Use FUFP tree maintenance approach to store only frequent sequential patterns to build FUSP tree.
2008	FUSP tree with pre-large [Tzung-Pei]	Use FUFP tree maintenance approach to build FUSP tree along with the concept of pre-large sequences.
2009	BSPinc [Lin et al.]	Backward mining strategy to eliminate the stable sequences whose support count does not change in the updated database.
2010	SPAMS [Vincaslas et al.]	Used for sequential patterns mining in data streams maintaining automation based structure for frequent sequential patterns.
2010	Stree_PS [Liu et al.]	Maintain sequence tree as the storage structure to keep all sequences of the original database along with their supports.
	IMSP [Hao et al.]	Generates a closed maximal frequency data model for the original sequence database instead of generating candidate sets.
2010	IPCI [Bisaria et al.]	Based on partitioning concept on theory of rough sets.
2011	Con_FST [Liu et al.]	Uses the frequent sequence tree as the storage structure to keep sequential patterns along with their supports.
2011	Algorithm using Pre- large sequences.	Based on the pre-large sequence concept to avoid the rescanning of the database till a defined safety bound.
2011	ICs pan [Yang et al.]	It uses sliding window model for data sampling and closed sequential pattern mining.

of sequential pattern is small, like to study biological/medical application where protein, DNA size is restricted or in case of study of disease with limited symptoms.

FASTUP, IUS, IncSP, FUSP-tree based algorithms, pre-large concept based algorithms generate efficient sequential patterns based on size of new transaction sequences added in the database. Applications, like study and analysis of software version change monitoring, where the databases do not grow continuously, these algorithms can be of more useful.

Table 4
Characteristics of various ISPM algorithms

ISPM algorithms	Huge memory overhead	Multiple scans of database	Huge candidate sets	Search space partition	Candidate sequence pruning	Depend on size of frequent patterns	Depend on size of new transaction sequences	Generate complete set of sequence patterns	Time consuming support counting	Effect of minimum support threshold	Use of negative border
Suffix tree	X					X		X			
FASTUP							X	X			
ISM	X					X		X			X
IUS		X	X			X	X	X	X		X
ISE		X	X						X		
KISP				X				X		X	
IncSpan	X			X	X	X			X		
IncSP		X	X		X		X	X	X		
Improved IncSpan				X				X			
MILE	X						X				
I Cs pan				X				X			
IncSPAM				X				X			
IncSpan+				X	X	X		X			
DSPID				X				X			
IMCS	X			X		X	X	X			
FUSP tree maintenance				X		X	X	X			
FUSP Tree with Pre-large				X			X	X			
BSPinc	X				X	X		X			
SPAMS	X					X				X	
Stree_PS				X				X		X	
IMSP				X				X		X	
IPCI				X		X		X			
Con_FST				X	X			X		X	
Pre- large sequence						X	X			X	

Stree_PS, IMSP, Con_FST, DSPID, IncSPAM algorithm are more suitable for applications where there is huge generation of sequences and continuous updating in the database. These algorithms do not have huge memory requirement, no multiple scanning of database is performed, either less or no candidate generation; search space is less and they do not have any impact of size of frequent/new transaction patterns. Market basket analysis, stock market analysis, web usage analysis are the applications, where these algorithms can contribute better than the other ISPM algorithms.

SPAMS, Stree_PS, IMSP, Con_FST, Pre-large sequence are the algorithms whose performance is affected by value of minimum support threshold. So, these algorithms are not suitable for applications with changing and possibly high threshold values, like source code mining/text mining where the user is interested in studying the text patterns with different values or for the study of customer purchase behavior with varying attributes that can act as support values, etc.

IUS, ISE, IncSP are the algorithms that require multiple scans of database and generate huge candidate sets. These algorithms are hence useful in applications where the numbers of database sequences are less. For example, to study intrusion detection in a very small network is one such application.

IncSpan, Improved IncSpan, IncSPAM, IncSpan+, DSPID, IMCS, FUSP-tree based algorithms, IMSP, IPCI are the algorithms, that would give fast mining result as they are based on partitioning the search space of the database. So they could be significant when time efficiency is more important, like in case of scientific applications, weather forecasting etc. However, IncSpan, IMCS has huge memory requirement so they could be useful only if the tradeoff between time and memory is feasible.

It has also been realized, that the ISE, IncSpan, SPAMS, Prelarge sequences based algorithms do not generate complete set of frequent sequential patterns. The analysis inferences and decision made on the basis of mining results given by these algorithms for any application would be incorrect and incomplete. These algorithms may even lead to loss of important information.

In general, the above discussion only gives an idea that how different ISPM algorithms' characteristics could be exploited for the use in any application. However, the selection of a particular ISPM algorithm or a combination of these algorithms in real-time, depends on various factors like performance parameters, hardware configuration, time and space requirement parameters etc as required for any application.

6. Limitations of ISPM algorithms

The ISPM algorithms, studied so far, have focused on two aspects:

1. How to deal with the incremental updating of sequential patterns when new data-sequences are added to the original database?
2. How to deal with the maintenance of sequential patterns when the minimum support threshold changes and there is no change in the original database?

But in the real time applications, like medical sciences, electronic commerce and web usage mining, we often delete some data from sequence database. This is done due to different reasons like, to save storage space, or may be some data is not interesting any longer, or may be it has become invalid.

The ISPM algorithms discussed in Section 4, could only handle insertion of sequences or appending items into the tail of each sequence. There are other types of modifications such as adding or removing items from a sequence or deletion of complete sequence. The ISPM algorithms cannot work in such cases. So, if a certain sequence does not have any newly arriving elements, this sequence will still stay in the database and undesirably contribute to count of the frequent sequential patterns. This generates high number of frequent sequences and as the deletion of the obsolete data from the sequence database is not

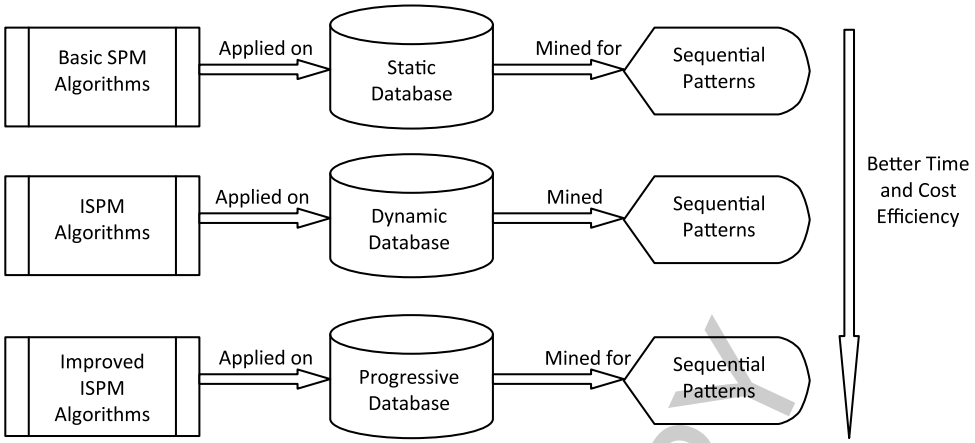

Fig. 14. Evolutions of SPM algorithms with time.

considered so many of these are uninteresting patterns. It is noted that users are usually more interested in the recent data than the old ones. Moreover, the new arriving patterns may not be identified as frequent sequential patterns, due to the existence of old sequences. The obsolete sequential patterns that are not frequent recently, may be present in the reported results. The sequential patterns generated with these ISPM algorithms, hence, give incomplete or inefficient patterns.

The solution comes in form of mining the sequential pattern from the progressive databases [1]. This type of mining not only adds new data to the original database but also removes obsolete data from the database. Hence, SPM could be classified into three categories:

- The SPM performed on static database finds the sequential patterns in the database that do not change over time.
- The ISPM done on an incremental database that corresponds to the mining process where there are new data arriving with the change in time i.e., the database is incremental.
- The Progressive Sequential Pattern Mining (PSPM) performed on a progressive database is the one where the new data are added into the database and obsolete data are removed simultaneously. Therefore, one can find the most up-to-date sequential patterns that are not influenced by obsolete data [22].

Thus, the SPM with a static database and with an incremental database can be considered as the special cases of the PSPM. When the obsolete data are not deleted from the database, the proposed approach for the PSPM is same as in case of ISPM. However, if the database does not have new data and does not delete obsolete data, then the PSPM approach can solve the static SPM problem. Figure 14, gives the diagrammatic representation of these three categories of SPM algorithms.

7. PSPM algorithms

This section makes study of those the algorithms that perform SPM on progressive database, which handle the new sequence addition besides deletion of obsolete data. These algorithms have been termed as PSPM algorithms henceforth in our discussion (Table 5). The three algorithms PISA, Weighted algorithm and CISpan has been discussed in detail with the illustrations, as we find these algorithms to address most of the limitation of the ISPM algorithms. However, other relevant work done for PSPM is also discussed, to give future direction for research.

Table 5
Concepts used by ISPM algorithms on progressive databases in their implementation

Year	ISPM algorithms on progressive databases	Theoretical Basis
2001	Pre-large sequence algorithm for record deletion [Wang et al.]	Use Pre-large sequences to maintain the database for customer sequences deletion.
2002	GSP+ [Zhang et al.]	Based on static database algorithm GSP and implement two pruning lemmas to reduce candidate in iterations.
2002	MFS+ [Zhang et al.]	Based on static database algorithm MFS and implement two pruning lemmas to reduce candidate in iterations.
2006	SPAM+ [Huang et al.]	Based on static database algorithm SPAM.
2006	MA_D [Ren and Zhou]	Maintains the sequential patterns obtain from the updating of database. It uses the information of previous mining results.
2006	IU_D [Ren and Zhou]	Maintains the incremental updating of the database with every deletion by using the information of previous mining results.
2008	PISA [Huang et al.]	Provides a general model to find the sequential pattern in defined period of interest (POI). It uses the sliding window concept. .and maintains tree to store latest data sequence.
2008	CISpan [Yuan et al.]	Used for closed sequential pattern by building tree- like structure, incremental lattice to store frequent sequences.
2009	Weighted approach to extract sequential pattern from progressive database [Mhatre et al.]	It provides weights to individual items or timestamps of sequential pattern so that the importance of the extracted sequential pattern from the progressive database could be measured.
2009	FUSP tree update algorithm [Lin, 2009]	An F USP tree is maintained to incrementally mine the progressive database for sequential patterns.

An algorithm [42] based on the concept of pre- large sequence is proposed to maintain the database with record deletion. Pre-large sequence as defined by a lower support threshold and an upper support threshold behave like a buffer [43]. This prevents the transition of sequential patterns directly from large to small and vice versa. The algorithm maintains the large and pre-large sequences with their counts from original database. The deleted customer sequences are compared with them. If found they are categorized as small, pre-large and large. Accordingly the support ratio is changed, otherwise no action is required. The algorithm does not require re-scanning the original database until the deleted customer sequences exceeds a threshold. This threshold value depends on database size. This proposed approach is hence more efficient for large databases.

Zhang et al. [32] developed two algorithms for PSPM, based on static algorithms one based on GSP, and the other based on MFS. To handle the deletion and the insertion of data, they employed two pruning lemmas that reduce the number of candidates with iterations. This further improves the scanning time of the database. However, MFS+ and to some extent GSP+ have to mine sequential patterns separately from each sub database for all combinations of Period of Interest (POIs). Thus, the performance improvement of GSP+ and MFS+ over GSP and MFS is not much significant.

MA_D (Maintenance Algorithm when Deleting some information) [14] algorithm utilizes the information obtained from previous mining processes. A new method was developed to generate the minimum sets of candidate sequences that can be used for further mining for better results. When some information is deleted from an original sequence database, MA_D algorithm performs incremental updating to discover frequent sequences from updated database with respect to the same minimum support threshold. This minimizes the overall runtime of the whole process. However, the multiple pass performed on the database is the limitation of the *MA_ D* algorithm.

Ren and Zhou [23] have presented an algorithm, called IU_D, for mining frequent sequences when some transactions and/or data sequences are deleted from the original sequence database. This algorithm

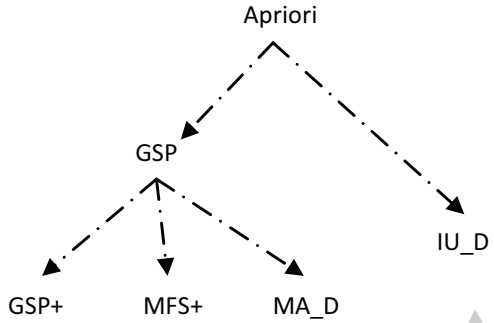

Fig. 15. The naïve approach used by ISPM algorithms that work on progressive databases.

also uses information obtained during earlier mining process to reduce the cost of finding new sequential patterns in the updated database.

Huang et al. employed a more recent method SPAM, as it is known as a better algorithm than other static sequential pattern algorithms. They modified this algorithm to SPAM+ and experimentally proved that the proposed algorithm outperforms the static SPAM algorithm with a considerable factor. Figure 15 gives the naïve approaches implemented by some of these PSPM algorithms.

FUSP-Tree Update Algorithm [4] uses FUSP- tree to maintain the incremental update in the sequences. A FUSP tree is built initially from the original database. Later the tree is maintained with every update and deletion of customer sequences. The FUSP tree is based on the concept of FUFP tree and IncSpan tree. The FUSP tree maintains the complete sequences that avoids and reduce the cost of rescanning the original database. This algorithm based on tree update give better result than the batch FUSP-tree algorithm (discussed in Section 4.3) to handle the deletion of customer sequences.

An efficient approach based on a general model of SPM in a progressive database, PISA (Progressive mIning of Sequential pAttern) [22] is proposed. It is based on the concept of sliding window which is continuously progressing along with time. The window length is specified by the user and corresponds to the particular timestamp called as POI (Period of Interest). Let us explain the concept behind Pisa algorithm with a sample database 'S' (Fig. 16). 'W' represents the POI with sequences of item/item set whose timestamp falls in the interval $[p, q]$, that is, $W^{p,q}$ is the window to represent time period starting with timestamp p and end with timestamp q. $T_1, T_2, T_3, T_4 \ldots T_{10}$ represent the timestamps and 101, 102, 103–106 are the sequence identifiers. a, b, c, d, e, f, g and h are the items in the database. Let the minimum support threshold, minSupp be 0.3 and the POI be five timestamps. The minimum frequency for any sequential pattern to be treated as frequent for our example would be 5×0.3 ($W^{p,q}$ $\times$ minSupp) $= 1.5$. So for $W^{1,5}$ in the database 'S' , we can discover 'ab' as the frequent sequential pattern, whose occurrence frequency is 3 as in sequence 101, 103, 105. However, for all the timestamps further till T_{10}, 'ab' is not a frequent pattern in any POI. The Pisa algorithm can delete this pattern at timestamp T_6 for being obsolete. Further, the algorithm incrementally discovers sequential patterns in defined time of POI. It utilizes a PS_tree (progressive sequential tree) to efficiently maintain the latest data sequences, discover the complete set of up-to-date sequential patterns, and delete obsolete data and patterns accordingly. PS_tree stores the element, timestamps of sequences in each POI and the occurrence frequency of every possible frequent sequential pattern at any time. The height of the sequential pattern tree depends on the length of POI. Hence, the memory requirement by Pisa is limited as compared to those by the alternative methods. It requires only one scan of the candidate sequences maintained by the PS-tree to discover frequent sequent pattern at each timestamp.

Seq ID	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
101	a	b	d	ca	e		b	f	bd	
102	d	b	f	a				c	d	
103	fa	b	c	df	cb		h		bc	
104		g	af	c	b				bc	
105		a	b		cb	e		f	h	
106			d	b	f		eb		b	d

Fig. 16. The example database 'S' to illustrate the PISA and Weighted algorithm.

Mhatre et al. [1] contribute further to this work, by proposing a method to prioritize between any two patterns mined at any instance of time. The method presented by them considers the time intervals between successive items, in addition to the item occurrence order. It is obvious, that an occurrence of a sequence with more time intervals in between its items would have lesser participation in determining the total support count for that sequence; as compared to an occurrence with no or lesser time intervals with the same POI. So this method provides a metric to measure the importance of the extracted sequential pattern from the progressive database. Using the same example database 'S', given in Fig. 16, the PISA algorithm for POI $W^{1,5}$ will give 'ab', 'bc' and 'df' as frequent sequential pattern with minSupp value as 0.3. On the contrary, this method would assign weight/priority to the item sets according to the time gaps within the items of the item sets. So, for our example, the sequence pattern 'bc' with a time gap of 2 timestamps in sequence 101 and 105 is not considered to be frequent sequential pattern by the weighted approach. On the other hand, it considers 'ab' as a frequent sequential pattern as the time gap between 'a' and 'b' is 1 in sequence 101, 103 and 105. The weighted approach maintains WM-ary tree, that is, Weighted M-ary tree to keep the information regarding, a) the item/itemset of the sequence, b) list of sequence identifiers, c) list of timestamps corresponding to each sequence, d) original timestamp that is the actual timestamp when the element occurs in a particular sequence and e) the time gap field to denote the number of inter-item time gaps in the pattern. The weighted algorithm insert, update and delete the nodes in the WM-ary tree as per the changes in the database. So, all candidate sequential patterns can be obtained from the tree for a particular POI.

CISpan, (Comprehensive Incremental Sequential PAtterN mining) [7] algorithm provides the solution in a divide and conquer manner. This separates the insertion and removal case apart from each other. The algorithm builds an incremental lattice, a tree-like data structure to store all the frequent sequences appearing in the inserted sequences. However, removal of sequences is done directly by updating the intermediate mining data structure, namely prefix lattice, of the original database. It is the compact representation of all the frequent sequence within the database. For incremental lattice, the number of nodes is confined only to items that are involved within the inserted sequences. However, the removed

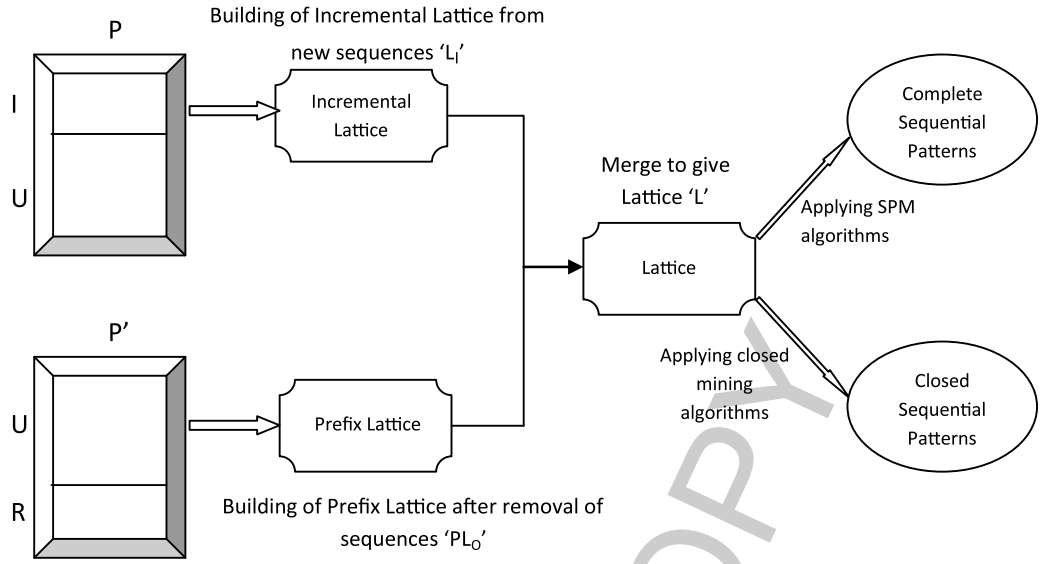

Fig. 17. Diagrammatic representation for the overview of CISpan.

sequences are handled by updating the original prefix lattice. CISpan algorithm then merges the original prefix lattice with the incremental lattice to give the updated lattice. A non- closed sequence elimination algorithm is applied to the updated lattice to give closed frequent sequences. The algorithm can also be used for generating complete frequent sequences. Let us illustrate this with an example. We use 'U' to represent the unchanged sequence set in our database, 'I' for the inserted sequence set and 'R' for the removed sequence set from the database (Fig. 17). 'P' is the updated database given by $P = U \cup I$ and P' is the database obtained after removal of sequences. The incremental lattice L_I is generated only from items in I. L_I calculate the support of sequences by using the database P. PL_O is the prefix lattice that is build from the original database. PL_O' is the prefix lattice derived from PL_O after the removal of sequences. The CISpan algorithm handle the insertion and removal of sequences in the database by performing the two activities of building L_I and PL_O' separately. It then merges these two lattices to give the updated lattice denoted as L. We can obtain frequent closed sequential pattern from this by applying closed sequential mining. The lattice L can also give complete frequent sequential pattern by performing basic SPM algorithms.

8. Applications of PSPM algorithms

As mentioned in Section 1, the mining of sequential patterns finds vast applications. With the discussion made so far, it is needless to state that, in practice with dynamic and distributed databases, the PSPM algorithms have more vital role to play. We demonstrate this, with some real-time applications as given below:

a) Text mining: There are tools like CP- Miner, Clospan, Mapo, Muvi, PR- Miner that tokenize the source text in certain ways. The tokens are then mapped to the sequence database. This database could be further mined to extract frequent sequential patterns using PSPM algorithms. Useful information, like programming rules, API usage, and copy-pasted code segments can be extracted.

For example, CP – Miner is a tool that detects copy-pasted text segments from the given text effectively. It first tokenizes every statement of the source text. Then it maps these tokens into a sequence database. When this database is mined for closed sequential patterns say with minimum support threshold of 3, it provides details of all copy pasted text segments that were repeated twice.

b) Stock Market analysis: Usually as per the local and global market scenarios, as well as economic condition of a state, stock market follows a particular trend in sales and purchase of stocks. These are the sequential patterns which could be mapped to our sequence database, which is dynamic in nature. This database, mined by using PSPM algorithms, could provide useful information like, which type of stocks suffers from major losses in particular situation, or in making decisions that purchase of which stock could book profit, based on past history.
c) Medical/Biological based application: Nowadays, there is an increase in the amount of biological data sequences, either DNA or protein sequences. These sequences once mapped to the sequence database could be mined based on their structure, function and role in chromosome. For example, one of the common problems that occur in analysis of biological sequences is to find the similarity in sequences of related proteins or genes. Such similarity (frequent sequential pattern) is due to the residues that are conserved during evolution of mankind. PSPM algorithm applied in this area can give desired results.
d) Retail Industry: The study of customer purchasing behavior is an example of modeling the behavior of some entity along time. By using a database, with transactions performed by customers at any instant, helps to predict what would be the customer's next transaction, based on his past transactions. This knowledge could help in decision making for sales and production of customer goods. This may even help in demand forecasting, inventory management, supply chain management and so on.
e) World Wide Web (WWW) application: There is a huge volume of daily log data collected on the server sites. This includes the sequences of page accesses by thousands of users on the web. This data (server logs) can be studied and analyzed to discover frequent user access patterns on WWW. We can use this knowledge to improvise the system design of the web application, like maintaining better hyperlinked structure between correlated pages. Going further, this could even help in making better marketing decisions by strategically placing the advertisement on the web.

9. Conclusion

Since they were defined in 1995, sequential patterns have received a great deal of attention. Earlier work in this area were focused on improving the efficiency of the algorithms, either with new structures, new representations or by managing the database in the main memory.

Validity of discovered patterns may change and new patterns may emerge due to updates on databases. In order to keep the current and up-to date sequential patterns, re-execution of the mining algorithm on the whole updated database is required. However, it takes more time than required in mining with the original database because of the new data sequences appended.

Therefore, the ISPM has gain popularity which utilizes previously discovered knowledge to solve the maintenance problem of the dynamic updated database efficiently without re-mining from scratch. Incremental mining is, however, not effective when the databases should be considered for updated information without the obsolete data. Under such situations the PSPM has become the obvious choice.

We can conclude that in present scenario there is a lot of research scope to be done in the area of ISPM applied on progressive databases. We are also working in this area; to give algorithms that are based on constraint based PSPM.

References

[1] A. Mhatre, M. Verma and D. Toshniwal, Extracting sequential patterns from progressive databases: A weighted approach, in: *IEEE International Conference on Signal Processing Systems* (2009), 788–792.

[2] C. Antunes and A.L. Oliveira, Sequential pattern mining algorithms: Trade-offs between speed and memory, *Second International Workshop on Mining Graphs, Trees and Sequences*, Pisa, Italy (2004).

[3] C. Luo and S.M. Chung, Efficient mining of maximal sequential patterns using multiple samples, in: *Proceedings of SDM* (2005), 64–72.

[4] C.W. Lin, T.P. Hong and W.H. Lu, An efficient FUSP-tree update algorithm for deleted data in customer sequences, in: *Fourth International Conference on Innovative Computing, Information and Control* (2009), 1491–1449.

[5] C.W. Lin, T.P. Hong, W.H. Lu and W.Y. Lin, An incremental FUSP-tree maintenance algorithm, in: *Eighth International Conference on Intelligent Systems Design and Applications* (2008), 445–449.

[6] C.C. Ho, H.F. Li, F.F. Kuo and S.Y. Lee, Incremental mining of sequential patterns over a stream sliding window, in: *Sixth IEEE International Conference on Data Mining Workshop* Hong Kong, China, (2006), 677–681.

[7] D. Yuan, K. Lee, H. Cheng, G. Krishna, Z. Li, X. Ma, Y. Zhou and J. Han, CISpan: Comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining, in: *SIAM International Conference on Data Mining – SDM* (2008), 84–95.

[8] F. Masseglia, F. Cathalat and P. Poncelet, The PSP approach for mining sequential patterns, in: *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'98)* Nantes, France, (1998), 176–184.

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

[10] G. Chen, X. Wu and X. Zhu, Sequential pattern mining in multiple streams, in: *Proceedings of the* 5^{th} *IEEE International Conference on Data Mining*, Houston, USA (2005), 27–30.

[11] H. Cheng, X. Yan and J. Han, IncSpan: Incremental mining of sequential patterns in large database, in: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2004), 527–532.

[12] J. Ayres, J. Gehrke, T. Yiu and J. Flannick, Sequential pattern mining using a bitmap representation, in: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), 429–435.

[13] J. Bisaria, N. Srivastava and K.R. Pardasani, A rough sets partitioning model for incremental mining of sequential patterns in large databases, *Applied Mathematical Sciences* **4** (2010), 1869–1881.

[14] J.D. Ren and X.L. Zhou, An efficient algorithm for incremental mining of sequential patterns, *Lecture Notes in Computer Science, Springer* **3930** (2006), 179–188.

[15] J.D. Ren and Y. Li, Mining weighted closed sequential patterns in large databases, *IEEE Fifth International Conference on Fuzzy Systems and Knowledge Discovery* (2008), 640–644.

[16] J. Han, H. Cheng, D. Xin and X. Yan, Frequent pattern mining; current status and future directions, *Data Mining and Knowledge Discovery, Springer* **15** (2007), 55–86.

[17] J. Han, J. Pei and X. Yan, Sequential pattern mining by pattern-growth: principles and extensions, *Springer-Verlag Berlin Heidelberg*, StudFuzz **180** (2005), 183–220.

[18] J. Li, B. Yang and W. Song, A new algorithm for mining weighted closed sequential pattern, in: *IEEE Second International Symposium on Knowledge Acquisition and Modeling* (2009), 338–341.

[19] J. Liu, S. Yan and J. Ren, The design of frequent sequence tree in incremental mining of sequential patterns, in: *IEEE* $2^{n.d}$ *International Conference on Software Engineering and Service Science (ICSESS)* (July 2011), 679–682

[20] J. Liu, S. Yan and J. Ren, The design of storage structure for sequence in incremental sequential pattern mining, in: *Sixth International Conference on Networked Computing and Advanced Information Management (NCM)* (2010), 330–334.

[21] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M.C. Hsu, Mining sequential patterns by pattern-growth: The PrefixSpan approach, *IEEE Transactions on Knowledge and Data Engineering* **16**(11) (2004), 1424–1440.

[22] J.W. Huang, C.Y. Tseng, J.C. Ou and M.S. Chen, A general model for sequential pattern mining with a progressive database, *IEEE Transactions on Knowledge and Data Engineering* **20**(9) (2008), 1153–1167.

[23] J.D. Ren and X.L. Zhou, A new incremental updating algorithm for mining sequential patterns, *Journal of Computer Science* **2**(4) (2006), 318–321.

[24] K. Wang and J. Tan, Incremental discovery of sequential patterns, in: *Proceedings of ACM SIGMOD'96 Data Mining Workshop* Montreal, Canada, (1996), 95–102.

[25] L. Chang, D. Yang, T. Wang and S. Tang, IMCS: Incremental mining of closed sequential patterns, *Advances in Data and Web Management Lecture Notes in Computer Science* **4505** (2007), 50–61.

[26] L. Vinceslas, J.E. Symphor, A. Mancheron and P. Poncelet, SPAMS: A novel incremental approach for sequential pattern mining in data streams, *Advances in Knowledge Discovery and Management* **292** (2010), 201–216.

[27] M.J. Zaki, SPADE: An efficient algorithm for mining frequent sequences, *Machine Learning* **42**(1–2) (2001), 31–60.

[28] M. Lin and S. Lee, Incremental update on sequential patterns in large databases, in: *Proceedings of the Tools for Artificial Conference (TAI'98)* (1998), 24–31.
[29] M.Y. Lin and S.Y. Lee, Incremental update on sequential patterns in large databases by implicit merging and efficient counting, *Information Systems* **29**(5) (2004), 385–404.
[30] M.Y. Lin and S.Y. Lee, Interactive sequence discovery by incremental mining, *An International Journal of Information Sciences-Informatics and Computer Science* **165**(3–4) (October 2004), 187–205.
[31] M.Y. Lin, S.C. Hsueh and C.C. Chan, Incremental discovery of sequential patterns using a backward mining approach, in: *IEEE International Conference on Computational Science and Engineering* **1** (2009), 64–70.
[32] M. Zhang, B. Kao, D.W. Cheung and K.Y. Yip, Efficient algorithms for incremental update of frequent sequences, in: *Proceeding of Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '02)* (2002), 186–197.
[33] N.P. Lin, W.H. Hao, H.J. Chen, E. Hao and C.C. Chang, Discover sequential patterns in incremental database, *International Journal of Computers* (2007), 196–201.
[34] Q. Zhao and S.S. Bhowmick, Sequential pattern mining: A survey, Technical Report, CAIS, Nanyang Technological University, Singapore, 2003, p. 118.
[35] Q. Zheng, K. Xu, S. Ma and W. Lv, The algorithms of updating sequential patterns, in: *Proceeding of 5th International Workshop on High Performance Data Mining (HPDM), in Conjunction with 2nd SIAM Conference on Data Mining* USA, 2002.
[36] R. Agrawal and R. Srikant, Mining sequential patterns, in: *Proceedings of the* 11^{th} *International Conference on Data Engineering* Taipei, Taiwan (1995), 3–14.
[37] R. Agrawal, T. Imielinski and A. Swami, Database mining: A performance perspective, *IEEE Transaction on Knowledge and Data Engineering* **5**(6) (1993), 914–925.
[38] R. Srikant and R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)* Avignon, France, (1996), 3–17.
[39] S.N. Nguyen, X. Sun and M.E. Orlowska, Improvements of IncSpan: Incremental mining of sequential patterns in large database, *Lecture Notes in Computer Science* **3518** (2005), 442–451.
[40] S. Parthasarathy, M. Zaki, M. Ogihara and S. Dwarkadas, Incremental and interactive sequence mining, in: *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99)* Kansas City, MO, USA, (1999), 251–258.
[41] S.Y. Yang, C.M. Chao, P.Z. Chen and C.H. Sun, Incremental mining of closed sequential patterns in multiple data streams, *Journal of Networks* **6**(5) (2011), 728–735.
[42] T.P. Hong, C.Y. Wang and S.S. Tseng, An incremental mining algorithm for maintaining sequential patterns using pre large sequences, *Expert Systems with Applications* (2001), 7051–7058.
[43] T.P. Hong, C.Y. Wang and Y.H. Tao, A new incremental data mining algorithm using pre-large item sets, *Intelligent Data Analysis* (2001), 111–129.
[44] T.P. Hong, H.Y. Chen, C.W. Lin and S.T. Li, Incrementally fast updated sequential patterns trees, in: *Proceedings of the Seventh International Conference on Machine Learning and Cybernetics* Kunming (July 2008), 3991–3996.
[45] W.C. Liao, D.L. Yang, J.Wu and M.C. Hung, Fast and effective generation of candidate-sequences for sequential pattern mining, in: *IEEE Fifth International Joint Conference on INC, IMS and IDC* (2009), 2006–2009.
[46] W. Cui and H. Zhong, Discovering interesting sequential pattern in large sequence database, in: *IEEE Second Asia-Pacific Conference on Computational Intelligence and Industrial Applications* (2009), 270–273.
[47] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh et al., Top 10 algorithms in data mining, *Knowledge Information System, Springer-Verlag* London **14** (2008), 1–37.
[48] Y. Chen, J. Guo, Y. Wang, Y. Xiong and Y.Y. Zhu, Incremental mining of sequential patterns using prefix tree, *Lecture Notes in Computer Science* Springer **4426** (2007), 433–440.
[49] Z. Zhang, L. Zhang, S. Zhong and J. Guan, A new algorithm for mining sequential patterns, *IEEE Fifth International Conference on Fuzzy Systems and Knowledge Discovery* (2008), 625–629.