Sequential Pattern Mining: Survey and Current Research Challenges

Chetna Chand, Amit Thakkar, Amit Ganatra

***Abstract*— The concept of sequence Data Mining was first introduced by Rakesh Agrawal and Ramakrishnan Srikant in the year 1995. The problem was first introduced in the context of market analysis. It aimed to retrieve frequent patterns in the sequences of products purchased by customers through time ordered transactions. Later on its application was extended to complex applications like telecommunication, network detection, DNA research, etc. Several algorithms were proposed. The very first was Apriori algorithm, which was put forward by the founders themselves. Later more scalable algorithms for complex applications were developed. E.g. GSP, Spade, PrefixSpan etc. The area underwent considerable advancements since its introduction in a short span.**

In this paper, a systematic survey of the sequential pattern mining algorithms is performed. This paper investigates these algorithms by classifying study of sequential pattern-mining algorithms into two broad categories. First, on the basis of algorithms which are designed to increase efficiency of mining and second, on the basis of various extensions of sequential pattern mining designed for certain application. At the end, comparative analysis is done on the basis of important key features supported by various algorithms and current research challenges are discussed in this field of data mining.

***Keywords*— Sequential Pattern, Sequence Database, Itemsets, Apriori.**

I. INTRODUCTION

Sequential Pattern Mining finds interesting sequential patterns among the large database. It finds out frequent subsequences as patterns from a sequence database. With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining sequential patterns from their database. Sequential pattern mining is one of the most well-known methods and has broad applications including web-log analysis, customer purchase behavior analysis and medical record analysis. In the retailing business, sequential patterns can be mined from the transaction records of customers. For example, having bought a notebook, a customer comes back to buy a PDA and a WLAN card next time. The retailer can use such information for analyzing the behavior of the customers, to understand their interests, to satisfy their demands, and above all, to predict their needs.

In the medical field, sequential patterns of symptoms and diseases exhibited by patients identify strong symptom/disease correlations that can be a valuable source of information for medical diagnosis and preventive medicine. In Web log analysis, the exploring behavior of a user can be extracted from member records or log files. For example, having viewed a web page on "Data Mining", user will return to explore "Business Intelligence" for new information next time. These sequential patterns yield huge benefits, when acted upon, increases customer royalty.

A. Basic Concepts of Sequential Pattern Mining

1. Let $I = \{x_1, \ldots, x_n\}$ be a set of *items*, each possibly being associated with a set of *attributes*, such as value, price, profit, calling distance, period, etc. The value on attribute A of item x is denoted by $x.A$. An *itemset* is a non-empty subset of items, and an itemset with k items is called a *k-itemset*.
2. A *sequence* $\alpha = \langle X_1 \cdot \cdot \cdot X_l \rangle$ is an ordered list of itemsets. An itemset X_i $(1 \le i \le l)$ in a sequence is called a *transaction*, a term originated from analyzing customers' shopping sequences in a transaction database. A transaction Xi may have a special attribute, *time-stamp*, denoted by $Xi.time$, which registers the time when the transaction was executed. For a sequence $\alpha = \langle X_1 \cdot \cdot \cdot X_l \rangle$, we assume $Xi.time < Xj.time$ for $1 \le i < j \le l$.
3. The number of transactions in a sequence is called the *length* of the sequence. A sequence with length l is called an *l-sequence*. For an l-sequence α, we have $len(\alpha) = l$. Furthermore, the i-th itemset is denoted by $\alpha[i]$. An item can occur at most once in an itemset, but can occur multiple times in various itemsets in a sequence.
4. A sequence $\alpha = \langle X_1 \ldots X_n \rangle$ is called a *subsequence* of another sequence $\beta = \langle Y_1 \ldots Y_m \rangle$ $(n \le m)$, and β a *super-sequence* of α, if there exist integers $1 \le i_1 < \ldots < i_n \le m$ such that $X_1 \subseteq Y_{i1}, \ldots, X_n \subseteq Y_{in}$.
5. A *sequence database SDB* is a set of 2-tuples *(sid, α)*, where *sid* is a *sequence-id* and α a sequence. A tuple *(sid, α)* in a sequence database *SDB* is said to *contain* a sequence γ if γ is a subsequence of α. The number of tuples in a

Manuscript Received February 09, 2012.

Chetna Chand, Department of Computer Engineering, Charotar Institute of Technology (Faculty of Technology and Engineering), Charotar University of Technology Changa, Anand, Gujarat, India, 388421 chetnachand.it@ecchanga.ac.in

Amit Thakkar, Department of Information Technology, Charotar Institute of Technology (Faculty of Technology and Engineering), Charotar University of Technology Changa, Anand, Gujarat, India, 388421 amitthakkar.it@ecchanga.ac.in

Amit Ganatra, Department of Computer Engineering, Charotar Institute of Technology (Faculty of Technology and Engineering), Charotar University of Technology Changa, Anand, Gujarat, India, 388421 amitganatra.ce@ecchanga.ac.in

sequence database *SDB* containing sequence γ is called the *support* of γ, denoted by *sup* (γ). Given a positive integer *min_sup* as the *support threshold*, a sequence γ is a *sequential pattern* in sequence database *SDB* if *sup* (γ) $\geq$ *min_sup*. The *sequential pattern mining* problem is to find the *complete* set of sequential patterns with respect to a given sequence database *SDB* and a support threshold *min_sup*.

II. CLASSIFICATION OF SEQUENTIAL PATTERN MINING ALGORITHM

As described by Yen-Liang Chen and Ya-Han Hu [16] in recent years, many approaches in sequential pattern mining have been proposed, these studies cover a broad spectrum of issues. In general, there are two main research issues in sequential pattern mining.

1. The first is to improve the efficiency in sequential pattern mining process while the other one is to
2. Extend the mining of sequential pattern to other time-related patterns.

A. Improve the Efficiency by Designing Novel Algorithms

According to previous research done in the field of sequential pattern mining, Sequential Pattern Mining Algorithms mainly differ in two ways [14]:

(1) The way in which candidate sequences are generated and stored. The main goal here is to minimize the number of candidate sequences generated so as to minimize I/O cost.

(2) The way in which support is counted and how candidate sequences are tested for frequency. The key strategy here is to eliminate any database or data structure that has to be maintained all the time for support of counting purposes only.

Based on these criteria's sequential pattern mining can be divided broadly into two parts:

- Apriori Based
- Pattern Growth Based

1. Apriori-Based Algorithms

The Apriori [Agrawal and Srikant 1994] and AprioriAll [Agrawal and Srikant 1995] set the basis for a breed of algorithms that depend largely on the apriori property and use the Apriori-generate join procedure to generate candidate sequences. The apriori property states that "All nonempty subsets of a frequent itemset must also be frequent". It is also described as antimonotonic (or downward-closed), in that if a sequence cannot pass the minimum support test, its entire super sequences will also fail the test.

Key features of Apriori-based algorithm are: [14]

(1) Breadth-first search: Apriori-based algorithms are described as breath-first (level-wise) search algorithms because they construct all the k-sequences, in k^{th} iteration of the algorithm, as they traverse the search space.

(2) Generate-and-test: This feature is used by the very early algorithms in sequential pattern mining. Algorithms that depend on this feature only display an inefficient pruning method and generate an explosive number of candidate sequences and then test each one by one for satisfying some user specified constraints, consuming a lot of memory in the early stages of mining.

(3) Multiple scans of the database: This feature entails scanning the original database to ascertain whether a long list of generated candidate sequences is frequent or not. It is a very undesirable characteristic of most apriori-based algorithms and requires a lot of processing time and I/O cost.

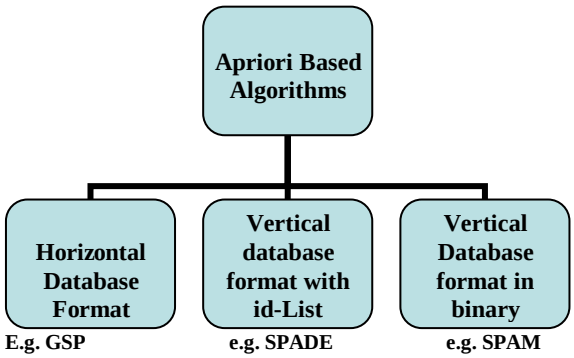

Fig. 1: Classification of Apriori based mining algorithm

i. GSP: The GSP algorithm described by Agrawal and Shrikant [2] makes multiple passes over the data. This algorithm is not a main-memory algorithm. If the candidates do not fit in memory, the algorithm generates only as many candidates as will fit in memory and the data is scanned to count the support of these candidates. Frequent sequences resulting from these candidates are written to disk, while those candidates without minimum support are deleted. This procedure is repeated until all the candidates have been counted. As shown in Fig 2, first GSP algorithm finds all the length-1 candidates (using one database scan) and orders them with respect to their support ignoring ones for which support < min_sup. Then for each level (i.e., sequences of length-k), the algorithm scans database to collect support count for each candidate sequence and generates candidate length (k+1) sequences from length-k frequent sequences using Apriori. This is repeated until no frequent sequence or no candidate can be found.

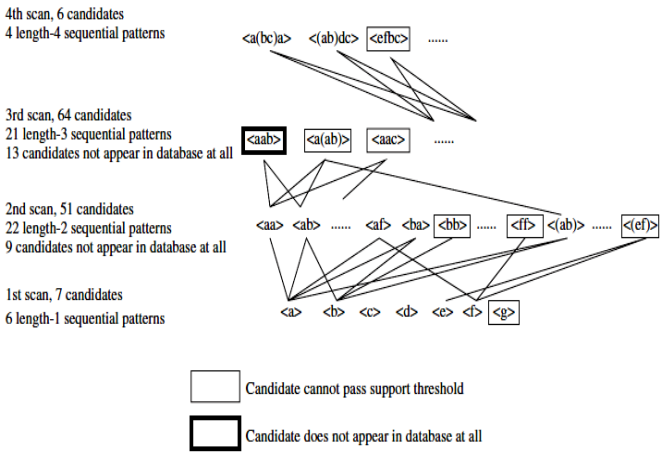

Fig. 2: Candidates, Candidate generation and Sequential Patterns in GSP

ii. SPIRIT: The Novel idea of the SPIRIT algorithm is to use regular expressions as flexible constraint specification tool [4]. It involves a generic user-specified regular expression constraint on the mined patterns, thus enabling considerably versatile and powerful restrictions. In order to push the constraining inside the mining process, in practice the algorithm uses an appropriately relaxed, that is less restrictive, version of the constraint. There exist several versions of the algorithm, differing in the degree to which the constraints are enforced to prune the search space of pattern during computation. Choice of regular expressions (REs) as a constraint specification tool is motivated by two important factors. First, REs provide a simple, natural syntax for the succinct specification of families of sequential patterns. Second, REs possess sufficient expressive power for specifying a wide range of interesting, non-trivial pattern constraints.

iii. SPADE: Besides the horizontal formatting method (GSP), the sequence database can be transformed into a vertical format consisting of items' id-lists. The id-list of an item as shown in fig 3, is a list of (*sequence-id*, *timestamp*) pairs indicating the occurring timestamps of the item in that sequence. Searching in the lattice formed by id-list intersections, the *SPADE* (Sequential Pattern Discovery using Equivalence classes) algorithm presented by M.J.Jaki [7] completes the mining in three passes of database scanning. Nevertheless, additional computation time is required to transform a database of horizontal layout to vertical format, which also requires additional storage space several times larger than that of the original sequence database.

Seq. ID	Sequence
1	< a (abc) (ac) d (cf) >
2	(ad) c (bc) (ae)
3	< (ef) (ab) (df) c b >
4	< e g (a f) c b c >

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	abc
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		....
SID	EID	SID	EID	....
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	3	
2	4	4	5	
3	2			
4	3			

ab			ba			...
SID	EID (a)	EID (b)	SID	EID (b)	EID (a)	...
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				...
SID	EID (a)	EID (b)	EID (a)	...
1	1	2	3	
2	1	3	4	

Fig. 3: Working of SPADE algorithm

iv. SPAM: SPAM integrates the ideas of GSP, SPADE, and FreeSpan. The entire algorithm with its data structures fits in main memory, and is claimed to be the first strategy for mining sequential patterns to traverse the lexicographical sequence tree in depth-first fashion. SPAM traverses the sequence tree in depth-first search manner and checks the support of each sequence-extended or itemset-extended child against min_sup recursively for efficient support-counting SPAM uses a vertical bitmap data structure representation of the database as shown in fig 4,which is similar to the id list in SPADE. SPAM is similar to SPADE, but it uses bitwise operations rather than regular and temporal joins. When SPAM was compared to SPADE, it was found to outperform SPADE by a factor of 2.5, while SPADE is 5 to 20 times more space-efficient than SPAM, making the choice between the two a matter of a space-time trade-off. [10]

CID	TID	Itemset
1	1	{a, b, d}
1	3	{b, c, d}
1	6	{b, c, d}
2	2	{b}
2	4	{a, b, c}
3	5	{a, b}
3	7	{b, c, d}

Dataset Sorted by CID and TID

CID	TID	{a}	{b}	{c}	{d}
1	1	1	1	0	1
1	3	0	1	1	1
1	6	0	1	1	1
-	-	0	0	0	0
2	2	0	1	0	0
2	4	1	1	1	0
-	-	0	0	0	0
-	-	0	0	0	0
3	5	1	1	0	0
3	7	0	1	1	1
-	-	0	0	0	0
-	-	0	0	0	0

Fig. 4: Transformation of Sequence database to Vertical binary format

2. Pattern-Growth Algorithms

Soon after the apriori-based methods of the mid-1990s, the pattern growth-method emerged in the early 2000s, as a solution to the problem of generate-and-test. The key idea is to avoid the candidate generation step altogether, and to focus the search on a restricted portion of the initial database. The search space partitioning feature plays an important role in pattern-growth. Almost every pattern-growth algorithm starts by building a representation of the database to be mined, then proposes a way to partition the search space, and generates as few candidate sequences as possible by growing on the already mined frequent sequences, and applying the apriori property as the search space is being traversed recursively looking for frequent sequences. The early algorithms started by using *projected databases*, for example, *FreeSpan* [Han et al. 2000], *PrefixSpan* [Pei et al. 2001], with the latter being the most influential.

Key features of pattern growth-based algorithm are: [14]

(1) Search space partitioning: It allows partitioning of the generated search space of large candidate sequences for efficient memory management. There are different ways to partition the search space. Once the search space is partitioned, smaller partitions can be mined in parallel. Advanced techniques for search space partitioning include projected databases and conditional search, referred to as split-and-project techniques.

(2) Tree projection: Tree projection usually accompanies pattern-growth algorithms. Here, algorithms implement a physical tree data structure representation of the search space, which is then traversed breadth-first or depth-first in search of frequent sequences, and pruning is based on the apriori property.

(3) Depth-first traversal: That depth-first search of the search space makes a big difference in performance, and also helps in the early pruning of candidate sequences as well as mining of closed

sequences [Wang and Han 2004]. The main reason for this performance is the fact that depth-first traversal utilizes far less memory, more directed search space, and thus less candidate sequence generation than breadth-first or post-order which are used by some early algorithms.

(4) Candidate sequence pruning: Pattern-growth algorithms try to utilize a data structure that allows them to prune candidate sequences early in the mining process. This result in early display of smaller search space and maintain a more directed and narrower search procedure.

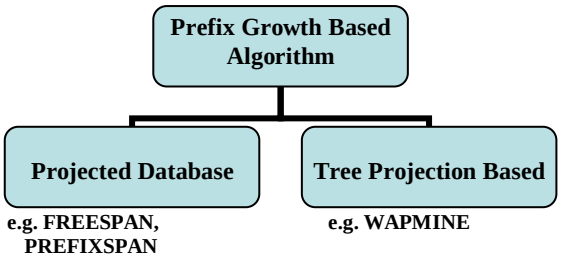

Fig 5: Classification of Prefix Growth based mining algorithm

i. FREESPAN: FreeSpan [5] was developed to substantially reduce the expensive candidate generation and testing of Apriori, while maintaining its basic heuristic. In general, FreeSpan uses frequent items to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database. Each projection partitions the database and confines further testing to progressively smaller and more manageable units. The trade-off is a considerable amount of sequence duplication as the same sequence could appear in more than one projected database. However, the size of each projected database usually (but not necessarily) decreases rapidly with recursion.

ii. WAP-MINE: It is a pattern growth and tree structure-mining technique with its WAP-tree structure. Here the sequence database is scanned only twice to build the WAP-tree from frequent sequences along with their support; a "*header table*" is maintained to point at the first occurrence for each item in a frequent itemset, which is later tracked in a threaded way to mine the tree for frequent sequences, building on the suffix. The WAP-mine [6] algorithm is reported to have better scalability than GSP and to outperform it by a margin. Although it scans the database only twice and can avoid the problem of generating explosive candidates as in apriori-based methods, WAP-mine suffers from a memory consumption problem, as it recursively reconstructs numerous intermediate WAP-trees during mining, and in particular, as the number of mined frequent patterns increases. This problem was solved by the PLWAP algorithm [Lu and Ezeife 2003], which builds on the prefix using position- coded nodes.

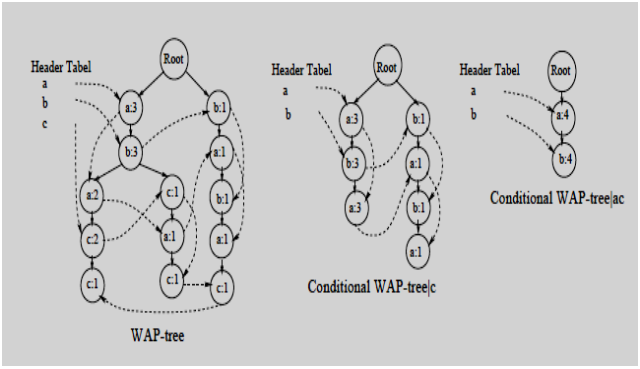

Fig. 6: Classification of Prefix Growth based mining algorithm

iii. PREFIXSPAN: The PrefixSpan (Prefix-projected Sequential pattern mining) algorithm presented by Jian Pei, Jiawei Han and Helen Pinto [8] representing the pattern-growth methodology, which finds the frequent items after scanning the sequence database once. The database is then projected as shown in Fig.7, according to the frequent items, into several smaller databases. Finally, the complete set of sequential patterns is found by recursively growing subsequence fragments in each projected database. Although the PrefixSpan algorithm successfully discovered patterns employing the divide-and-conquer strategy, the cost of memory space might be high due to the creation and processing of huge number of projected sub-databases.

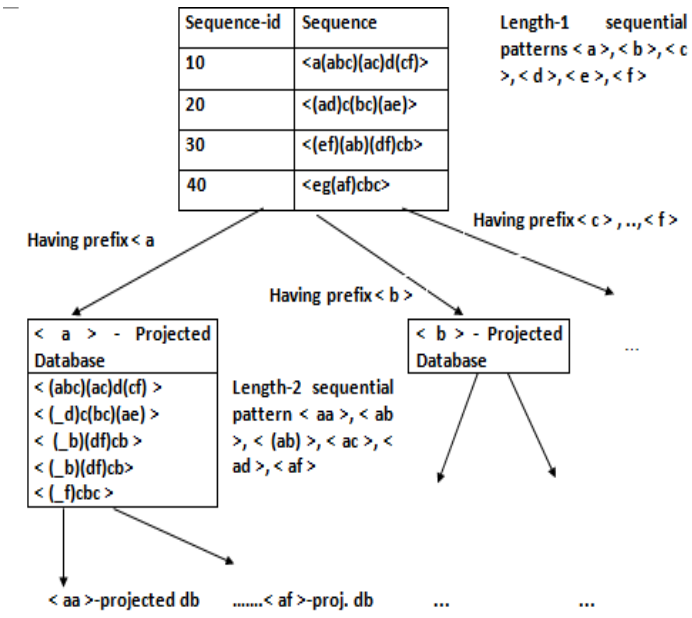

Fig. 7: Construction of Projected Databases in PrefixSpan Algorithm

B. Extensions of Sequential Pattern Mining to Other Time-Related Patterns

Sequential pattern mining has been intensively studied during recent years; there exists a great diversity of algorithms for sequential pattern mining. Along with that Motivated by the potential applications for the sequential patterns, numerous extensions of the initial definition have been proposed which may be

related to other types of time-related patterns or to the addition of time constraints. Some extensions of those algorithms for special purposes such as multidimensional, closed, time interval, and constraint based sequential pattern mining are discussed in following section.

i. Multidimensional Sequential Pattern Mining

Mining sequential patterns with single dimension means that we only consider one attribute along with time stamps in pattern discovery process, while mining sequential patterns with multiple dimensions we can consider multiple attributes at the same time. In contrast to sequential pattern mining in single dimension, mining multiple dimensional sequential patterns introduced by Helen Pinto and Jiawei Han [9] can give us more informative and useful patterns. For example we may get a traditional sequential pattern from the supermarket database that after buying product *a* most people also buy product *b* in a defined time interval. However, using multiple dimensional sequential pattern mining we can further find different groups of people have different purchase patterns. For example, M.E. students always buy product *b* after they buy product *a*, while this sequential rule weakens for other groups of students. Hence, we can see that multiple-dimensional sequential pattern mining can provide more accurate information for further decision support.

ii. Discovering Constraint Based Sequential Pattern

Although efficiency of mining the complete set of sequential patterns has been improved substantially, in many cases, sequential pattern mining still faces tough challenges in both effectiveness and efficiency. On the one hand, there could be a large number of sequential patterns in a large database. A user is often interested in only a small subset of such patterns. Presenting the complete set of sequential patterns may make the mining result hard to understand and hard to use. To overcome this problem Jian Pei, Jiawei Han and Wei Wang [13] have systematically presented the problem of pushing various constraints deep into sequential pattern mining using pattern growth methods.

Constraint-based mining may overcome the difficulties of effectiveness and efficiency since constraints usually represent user's interest and focus, which limits the patterns to be found to a particular subset satisfying some strong conditions. (Pei, Han, & Wang, 2007) mention seven categories of constraints:

1. Item constraint: An item constraint specifies subset of items that should or should not be present in the patterns.

2. Length constraint: A length constraint specifies the requirement on the length of the patterns, where the length can be either the number of occurrences of items or the number of transactions.

3. Super-pattern constraint: Super-patterns are ones that contain at least one of a particular set of patterns as sub-patterns.

4. Aggregate constraint: An aggregate constraint is the constraint on an aggregate of items in a pattern, where the aggregate function can be sum, avg, max, min, standard deviation, etc.

5. Regular expression constraint: A regular expression constraint CRE is a constraint specified as a regular expression over the set of items using the established set of regular expression operators, such as disjunction and Kleene closure.

6. Duration constraint: A duration constraint is defined only in sequence databases where each transaction in every sequence has a time-stamp. It requires that the sequential patterns in the sequence database must have the property such that the time-stamp difference between the first and the last transactions in a sequential pattern must be longer or shorter than a given period.

7. Gap constraint: A gap constraint set is defined only in sequence databases where each transaction in every sequence has a timestamp. It requires that the sequential patterns in the sequence database must have the property such that the timestamp difference between every two adjacent transactions must be longer or shorter than given gap.

Other Constraints:

R (Recency) is specified by giving a recency minimum support (r_minsup), which is the number of days away from the starting date of the sequence database. For example, if our sequence database is from 27/12/2007 to 31/12/2008 and if we set r_minsup = 200 then the recency constraint ensures that the last transaction of the discovered pattern must occur after 27/12/2007+200 days. In other words, suppose the discovered pattern is < (a), (bc)>, which means "after buying item a, the customer returns to buy item b and item c". Then, the transaction in the sequence that buys item b and item c must satisfy recency constraint. [17]

M (Monetary) is specified by giving monetary minimum support (m_minsup). It ensures that the total value of the discovered pattern must be greater than m_minsup. Suppose the pattern is < (a), (bc)>. Then we can say that a sequence satisfies this pattern with respect to the monetary constraint, if we can find an occurrence of pattern < (a), (bc)> in this data sequence whose total value must be greater than m_minsup. [17]

C (Compactness) constraint, which means the time span between the first and the last purchase in a customer sequence, must be within a user-specified threshold. This constraint can assure that the purchasing behavior implied by a sequential pattern must occur in a reasonable period. [17]

Target-Oriented A target-oriented sequential pattern is a sequential pattern with a concerned itemset in the end of pattern. For most decision makers, when they want to make efficient marketing strategies, they usually concern the happening order of a concerned itemsets only, and thus, most sequential patterns discovered by using traditional algorithms are irrelevant and useless. [18]

iii. Discovering Time-interval Sequential Pattern

Although sequential patterns can tell us what items are frequently bought together and in what order, they cannot provide information about the time span between items for further decision support. In other words, although we know which items will be

bought after the preceding items, we have no idea when the next purchase will happen. Y. L. Chen, M. C. Chiang, and M. T. Kao [11] have given the solution of this problem that is to generalize the mining problem into discovering time-interval sequential patterns, which tells not only the order of items but also the time intervals between successive items. An example of time-interval sequential pattern is (*a*, *I1*, *b*, *I2*, *c*), meaning that we buy item *a* first, then after an interval of *I1* we buy item *b*, and finally after an interval of *I2* we buy item *c*. Similar type of work done by C. Antunes, A. L. Oliveira, [8] by presenting the concept of gap constraint. A gap constraint imposes a limit on the separation of two consecutive elements of an identified sequence. This type of constraints is critical for the applicability of these methods to a number of problems, especially those with long sequence.

iv. Closed Sequential Pattern Mining

The sequential pattern mining algorithms developed so far have good performance in databases consisting of short frequent sequences. Unfortunately, when mining long frequent sequences, or when using very low support thresholds, the performance of such algorithms often degrades dramatically. This is not surprising: Assume the database contains only one long frequent sequence < (*a*1) (*a*2) . . . (*a*100) >, it will generate $2^{100}-1$ frequent subsequence if the minimum support is 1, although all of them except the longest one are redundant because they have the same support as that of < (*a*1) (*a*2) . . . (*a*100) > . So proposed an alternative but equally powerful solution: instead of mining the complete set of frequent subsequence, we mine frequent *closed subsequence* only, i.e., those containing no super-sequence with the same support.

This mining technique will generate a significant less number of discovered sequences than the traditional methods while preserving the same expressive power since the whole set of frequent subsequences together with their supports, can be derived easily from the mining results. [12]

III. COMPARITIVE STUDY OF SEQUENTIAL PATTERN MINING ALGORITHMS

Comparative analysis of sequential pattern mining algorithm is done on the basis of their various important features. For comparison sequential pattern mining is divided into two broad categories, namely, Apriori Based and Pattern Growth Based Algorithms. All the nine features used to classify these algorithms are discussed first and then comparison is done for the following algorithms:

GSP: Generalized Sequential Patterns

SPADE: Sequential Pattern Discovery using Equivalence classes

SPAM: Sequential Pattern Mining

FREESPAN: Frequent pattern projected Sequential pattern mining

PREFIXSPAN: Prefix-projected Sequential pattern mining

WAPMINE: Web Access Pattern Mining

SPIRIT*:* Sequential Pattern mining with Regular expression constraints

Characteristics of Sequential Pattern Mining Algorithm are:

Apriori-Based vs. Pattern-Growth-Based Apriori-based algorithms usually use a candidate "generate-and-test" type of approach ,which exploits the downward closure property: if an itemset α is not frequent, then any superset of α must not be frequent either, Pattern-growth algorithms take a more incremental approach in generating possible frequent sequences, and use what might be called a divide-and-conquer approach. Pattern-growth algorithms make projections of the database in an attempt to reduce the search space.

BFS-Based Approach Vs. DFS-Based Approach In a BFS approach level-by-level search can be conducted to find the complete set of patterns i.e. All the children of a node are processed before moving to the next level. On the other hand, when using a depth-first search approach, all sub-arrangements on a path must be explored before moving to the next one. The advantage of DFS over BFS is that DFS can very quickly reach large frequent arrangements and therefore, some expansions in the other paths in the tree can be avoided.

Top-Down Search Vs. Bottom-Up Search Apriori-based algorithms employ a bottom-up search, enumerating every single frequent sequence. This implies that in order to produce a frequent sequence of length l, all 2^l subsequences have to be generated. It can be easily deduced that this exponential complexity is limiting all the Apriori-based algorithms to discover only short patterns, since they only implement subset infrequency pruning by removing any candidate sequence for which there exists a subsequence that does not belong to the set of frequent sequences. In a top-down approach the subsets of sequential patterns can be mined by constructing the corresponding set of projected databases and mining each recursively from top to bottom.

Table 1: Comparative Study of Sequential Pattern Mining Algorithms

ALGO. / CHAR.	GSP	SPADE	SPAM	FREE-SPAN	PREFIX SPAN	WAP MINE	SPIRIT
APIRORI BASED	✓	✓	✓				✓
PATTERN GROWTH BASED				✓	✓	✓	
BFS-BASED APPROACH	✓						✓
DFS-BASED APPROACH		✓	✓	✓	✓	✓	
TOP-DOWN SEARCH				✓	✓	✓	
BOTTOM-UP SEARCH	✓	✓	✓				✓
ANTI-MONOTONE PROPERTY	✓	✓					✓
PREFIX-MONOTONE PRFOPERTY					✓		
REGULAR EXPRESSION CONSTRAINT				✓	✓	✓	✓

Anti-Monotone Vs. Prefix-Monotone Property Anti-Monotone property states that every non-empty sub-sequence of a sequential pattern is a sequential pattern, while Prefix-Monotone property states that if for each α sequence satisfying the constraint, so does every sequence having α as a prefix also satisfies the constraint.

Regular Expression Constraint Complexity of regular expression constraints can be roughly measured by the numbers of state changes in

their corresponding deterministic finite automata. A regular expression constraint has a nice property called growth-based anti-monotonic. A constraint is growth-based anti-monotonic if it has the following property: If a sequence satisfies the constraint must be reachable by growing from any component which matches part of the regular expression.

From the comparative study of table 1, it is clear that PrefixSpan algorithm uses depth first search based approach, top down search which are efficient techniques to find frequent subsequences as sequential patterns form the large database. Also PrefixSpan uses regular expression constraints as well as prefix monotone property, which makes this algorithm an obvious choice for applying user defined constraints for mining only some concerned sequential patterns.

A. Experimental Analysis Done By Researchers [16]

To evaluate the effectiveness and efficiency of various sequential pattern mining algorithms, an extensive performance study is performed on four algorithms: PrefixSpan, FreeSpan, GSP, and SPADE, on both real and synthetic data sets.

Dataset Detail

Synthetic Datasets is used for the performance study Synthetic dataset used in the experiment are C10T8S8I8, C200T2.5S10I1.25, C200T5S10I2.5 where,
C = Number of Customer
T = Avg. number of items / transaction
S = Avg. number of transaction / Sequence
I = Average itemset in maximal sequence
It is assumed that number of items is 10,000 and on average, a frequent sequential pattern consists of four transactions.

i. Comparison of Memory Usage

From the comparison graph of fig. 8, it is clear that PrefixSpan is not only more efficient, but also more stable in memory usage than both SPADE and GSP. At support 0.25 percent, GSP cannot stop running after it has consumed about 362 MB memories and SPADE reported an error message, while PrefixSpan only uses 108 MB memory. Based on the analysis, PrefixSpan only needs memory space to hold the sequence data sets plus a set of header tables and pseudo projection tables.

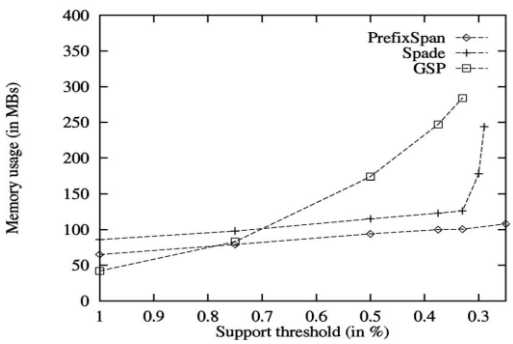

Fig. 8: Memory usage of algorithms on data set C200T5S10I2.5

ii. Comparison of Time Complexity

As from the comparison graph from fig. 9, it is clear that both of the pattern growth algorithm, Freespan and Prefixspan are time efficient than Apriori based Algorithm.

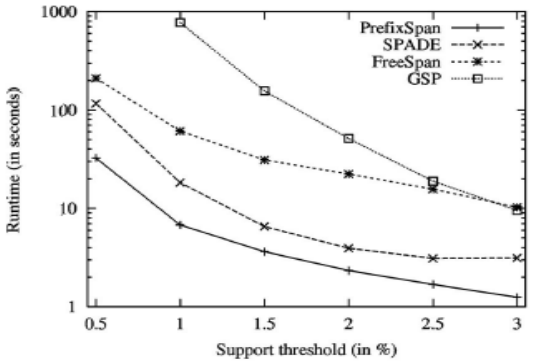

Fig.9: Performance of the four algorithms on data set C10T8S8I8

ii. Comparison of Scalability

From the Experimental results shown in fig. 10, it is clear that PrefixSpan is many time faster than other algorithms and scale linearly with increasing database sizes. Since PrefixSpan needs memory space to hold the sequence database plus a set of header tables and pseudoprojection tables on the other hand, both SPADE and GSP need memory space to hold candidate sequence patterns as well as the sequence databases.

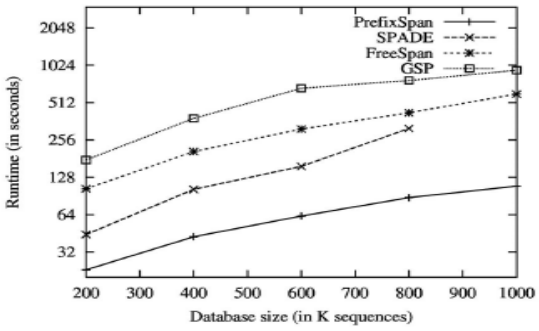

Fig.10: Scalability test of algorithms on data set T2:5S10I1:25, with min_support 0.5 percent.

From the above performance study it is clear that PrefixSpan is the clear winner among all the four tested algorithms. Reason for this high performance is discussed below:

- *Pattern-growth without candidate generation:*
  Unlike traditional Apriori-based approach which performs candidate generation-and test, PrefixSpan does not generate any useless candidate and it only counts the frequency of local 1-itemsets.
- *Projection-based divide-and-conquer as an effective means for data reduction:*
  PrefixSpan grows longer patterns from shorter ones by dividing the search space and focusing only on the subspace potentially supporting further pattern growth. The search space of PrefixSpan is focused and is confined to a set of projected databases. Since a projected database for a sequential pattern α contains all and only the necessary information for mining the sequential patterns that can grow

from α, the size of the projected databases usually reduces quickly as mining proceeds to longer sequential patterns. In contrast, the apriori-based approach always searches the original database for all iterations during mining process. Many irrelevant sequences have to be scanned and checked, which adds to the overhead. This argument is also supported by our performance study.

- *PrefixSpan consumes relatively stable memory space:* Since PrefixSpan generates no candidates and explores the divide-and-conquer methodology, it consumes stable memory space throughout the mining process. On the other hand, the candidate generation-and-test methods, including both GSP and SPADE, require a substantial amount of memory when the support threshold goes low since it needs to hold a tremendous number of candidate sets.

IV. LIMITATION OF EXISTING APPROCHES

1. In Apriori based algorithm, huge set of candidate sequences could be generated in a large sequence database. For example, if there are 1,000 frequent length-1 sequences then it will generate $1000 \times 1000 + \frac{1000 \times 999}{2} = 1{,}499{,}500$ length-2 candidates! [8]

2. Multiple scans of original database is required during the mining process in Apriori based Algorithms. [16]

3. The major cost of Pattern Growth based Algorithm is the construction of projected databases. In the worst case, they have to construct projected database for every sequential pattern. If there present a good number of sequential patterns, the cost is nontrivial. [8]

4. Difficulty in mining long sequential patterns: This is because a long sequential pattern must grow up from a huge number of short sequential patterns, but the number of such candidate sequences generated is exponential to the length of the sequential patterns to be mined. [16]

5. Use of frequency as the interestingness measures generates exponential number of sequential patterns. But a user is often interested in only a small subset of such patterns. Presenting the complete set of sequential patterns may make the mining result hard to understand and hard to use. [18]

V. RESEARCH CHALLENGES

Today several methods are available for efficiently discovering sequential patterns according to the initial definition. Such patterns are widely applicable for a large number of applications. But still there are various research challenges in this field of data mining. Some of the research challenges are:

- To find the complete set of patterns, when possible, satisfying the minimum support (frequency) threshold.
- To be highly efficient, scalable, involving only a small number of database scans.
- To be able to incorporate various kinds of user-specific constraints. [13]
- Algorithm should handle large search space.
- Algorithm should avoid repeated scanning of database during mining process.
- To study constraints like Recency, Frequency and Monetary constraints and check their effect with respect to execution time, memory usage and scalability.
- To add other useful constraints to the RFM patterns, for example, the constraint that the number of repetitions in a sequence must be no less than a given threshold. [19]
- To study target oriented sequential pattern mining and its application in some real dataset.[18]
- To use some method by which early candidate sequence pruning and search space partitioning will be possible for efficient mining of patterns.
- To introduce the concept of object-orientedness in sequential pattern mining, by which there will be flexibility of mining only, focused parts of the database.
- There are many interesting issues that need to be studied further, Especially, the developments of specialized sequential pattern mining methods for particular applications, such as DNA sequence mining that may admit faults, such as allowing insertions, deletions, and mutations in DNA sequences, and handling industry/engineering sequential process analysis are interesting issues for future research.[15]
- For large sequence database there can be a possibility of having distributed sequential pattern mining to provide scalability.
- Instead of using crisp (exact) constraint to discover patterns, it can be extended to using fuzzy constraints since; it is difficult for a retailer to appropriately set threshold value for the selection criteria.[17]
- Mining Multi-level Time-interval Sequential Patterns using fuzzy time value. [20]

VI. CONCLUSION

Sequence Data mining, the concept being introduced in 1995 has undergone considerable advancement in less than two decades. First work on this topic focused on improving the efficiency of the algorithms either with new structures, new representations or by managing the database in the main memory. So based on these criteria's sequential pattern mining is classified into two major groups, Apriori Based and Pattern Growth based algorithms. So, from the previous studies and comparative analysis of various mining algorithms, it is clear that pattern growth based algorithms are more efficient with respect to running time, space utilization and scalability.

ACKNOWLEDGMENT

Above all I would like to express my gratitude to the Almighty, who aided with his strength, wisdom and patience to complete this paper. I would like to acknowledge and extend my heartfelt gratitude to Mr. Amit Thakkar and Mr. Amit Ganatra for their valuable suggestion and support in the duration of study. I would like to thank my parents for their valuable support and inspiration. Last but not the least my special thanks go to our institute, CHARUSAT, for giving me this opportunity to work in the great environment.

REFERENCES

1. Rakesh Agrawal Ramakrishna Srikant, “Mining Sequential Patterns”, 11th Int. Conf. on Data Engineering, IEEE Computer Society Press, Taiwan, 1995 pp. 3-14.
2. Srikant R. and Agrawal R., “Mining sequential patterns: Generalizations and performance improvements”, Proceedings of the 5th International Conference Extending Database Technology, 1996, 1057, 3-17.
3. F. Masseglia, F. Cathala, and P. Poncelet, “The PSP Approach for Mining Sequential Pattern”, In Proc. 1998 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD’98), Nantes, France, Sept. 1998, pp. 176–184.
4. M. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential pattern mining with regular expression constraints", VLDB'99, 1999.
5. Han J., Dong G., Mortazavi-Asl B., Chen Q., Dayal U., Hsu M.-C., ”Freespan: Frequent pattern-projected sequential pattern mining”, Proceedings 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD’00), 2000, pp. 355-359.
6. Han, J., Pei, J., Mortazavi-Asl, B. and Zhu, H., “Mining access patterns efficiently from web logs”, In Proceedings of the Pacific- Asia Conference on Knowledge Discovery and Data Mining (PAKDD’00) Kyoto Japan, 2000.
7. M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences”, Machine Learning, 2001.
8. J. Pei, J. Han, B. Mortazavi-Asi, H. Pino, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix- Projected Pattern Growth", ICDE'01, 2001.
9. Helen Pinto Jiawei Han Jian Pei Ke Wang, “Multidimensional Sequential Pattern Mining”, In Proc. 2001 Int. Conf. Information and Knowledge Management (CIKM’01), Atlanta, GA, Nov. 2001 pp. 81–88.
10. AYRES, J., FLANNICK, J., GEHRKE, J., AND YIU, T., “Sequential pattern mining using a bitmap representation”, In Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining-2002.
11. Chen, Y.L., Chiang, M.C. and Ko, M.T, “Discovering time interval sequential patterns in sequence databases”, Expert Syst. Appl., Vol. 25, No. 3, 2003, pp. 343–354.
12. Yan, X., Han, J., and Afshar, R., “CloSpan: Mining closed sequential patterns in large datasets”, In Third SIAM International Conference on Data Mining (SDM), San Fransico, CA, 2003, pp. 166–177.
13. Jian Pei, Jiawei Han, Wei Wang, “Constraint-based sequential pattern mining: the pattern growth methods”, J Intell Inf Syst , Vol. 28, No.2, ,2007, pp. 133 –160.
14. NIZAR R. MABROUKEH and C. I. EZEIFE, ”A Taxonomy of Sequential Pattern Mining Algorithms”, ACM Computing Surveys, Vol. 43, No. 1, Article 3, Publication date: November 2010.
15. J. Han, J. Pei, and X. Yan, StudFuzz,”Sequential Pattern Mining by Pattern-Growth: Principles and Extensions”, 180, 2005, pp. 183–220.
16. J.Pei, J.Han, B.MortazaviAsl, J.Wang, H.Pinto, Q.Chen, U.Dayal and M.-C.Hsu, “Mining sequential patterns by pattern-growth: The PrefixSpan approach”, IEEE Transactions on Knowledge and Data Engineering, vol.16, no.11, 2004, pp. 1424-1440.
17. Yen-Liang Chen, Mi-Hao Kuo, Shin-Yi Wu, Kwei Tang, ”Discovering Recency, frequency, and monetary (RFM) sequential patterns from customers’ purchasing data”, Electronic Commerce Research and Applications 8 (2009), 2009, pp. 241–251.
18. Hao-En Chueh, “Mining Target-Oriented Sequential Patterns with Time-Interval”, International journal of computer science & information Technology (IJCSIT) Vol.2, No.4, August 2010.
19. Yen-Liang Chen, Ya-Han Hu, “The consideration of recency and compactness in sequential pattern mining”, In Proceedings of the second workshop on Knowledge Economy and Electronic Commerce, Vol. 42, Iss. 2 ,pp. 1203-1215, 2006.
20. Ya-Han Hu, Fan Wu, “Mining Multi-level Time-interval Sequential Patterns in Sequence Databases”, Chieh-I Yang, 2010.

AUTHOR PROFILE

Chetna Chand has received her B.E degree in Information Technology from Bhavnagar University, India in 2008 and currently pursuing her master degree in Computer Engineering from Charotar University of Science and Technology, Changa, Gujarat. She has 2 year of teaching experience and currently working as a lecturer in faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat. Her current research interest includes Data Mining, Sequential Pattern Mining.

Amit Thakkar has received his B.E degree in Information Technology from Gujarat University, Gujarat, India in 2002 and master Degree from Dharmsinh Desai University, Gujarat, India in 2007. He is currently pursuing his Ph.D in the area of Multi relational Classification at Kadi Sarvavishvidhalaya University, Gandhinagar, India in June 2010. Since 2002 he has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, Where he is currently working as an Associate Professor in the Department of Information Technology. He has published more than 20 research papers in the field of data mining and web technology. His current research interest includes Multi relational Data Mining, Relational Classification.

Amit Ganatra has received his B.E degree in Computer Engineering from Gujarat University, Gujarat, India in 2000 and master Degree from Dharmsinh Desai University, Gujarat, India in 2004. He is currently pursuing his Ph.D in Information Fusion Techniques in Data Mining at KSV University, Gandhinagar, Gujarat, India in August 2008. Since 2000 he has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat. He is concurrently holding Associate Professor (Jan 2010 till date), Headship in computer Engineering Department (since 2001 to till date) at CSPIT, CHARUSAT and Deanship in Faculty of Technology-CHARUSAT (since Jan 2011 to till date), Gujarat.

He is a member of Board of Studies (BOS), Faculty Board and Academic Council for CHARUSAT and member of BOS for Gujarat Technological University (GTU). He was the founder head of CE and IT departments of CITC (now CSPIT). His areas of interest include Database and Data Mining, Artificial Intelligence, System software, soft computing and software engineering. In these areas, he is having good research record and published and contributed over 70 papers (Author and Co-author) published in referred journals and presented in various international conferences. He has guided more than 90 industry projects at under graduate level and 47 dissertations at Post Graduate level. He has 11 years of teaching experience at UG level and concurrently 7 years of teaching and research experience at PG level, having good teaching and research interests.