Survey on Sequential Pattern Mining Algorithms

V.Chandra Shekhar Rao
Associate Professor
Computer Science & Engineering Dept
Kakatiya Institute of Technology & Science Warangal-506015, Andhra Pradesh, India

P.Sammulal,Ph.D
Assistant Professor
Computer Science & Engineering Dept
JNTU Nachupally
Karimnagar,(Dt),Andhra Pradesh, India

ABSTRACT

Sequential pattern mining is a significant data-mining method for determining time-related behavior in sequence databases. The information achieved from sequential pattern mining can be used in marketing, medical records, sales analysis, and so on. Existing methods only focus on the concept of frequency because of the assumption that sequences' behaviors do not change over time. Several efficient algorithms for maintaining sequential patterns have been developed. , old datasets are deleted while some other datasets are updated. It is obvious time stamp as an important attribute of each dataset, also it is important in the process of data mining and it can gives us more accurate and useful information. Although there have been many recent studies on the sequential patterns in static database. But the complexity of sequential pattern mining is when increasing the data in dynamically, As time passes by new data sets are inserted

Keywords

Sequential pattern mining, Sequence Database, Apriori, SPADE, Time constraint.

I. INTRODUCTION

Sequential pattern mining: Sequential pattern mining is an important subject of data mining, a further promotion of association rule mining, and it is also widely applied [1]. Sequential pattern mining algorithms address the problem of discovering the existent frequent sequences in a given database [2]. Sequential pattern mining is closely related to association rule mining, except that the events are linked by time [3]. Sequential patterns indicate the correlation between transactions while association rule represents intra transaction relationships. In association rule mining, the mining results are about which items are brought together frequently, those items must come from the same transaction. While the results of sequential pattern mining are about which items are bought in a certain order by the same customer, with those items coming from different transactions [4]. Sequential patterns can help managers determine which items are bought one after another in a sequence [5], or to analyze browsing orders of homepages in a Web site [6] and more.

Sequential pattern mining is commonly defined as finding the complete set of frequent subsequences in a set of sequences [7]. 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. Each sequence corresponds to a temporally ordered list of events, where each event is a collection of items (item set) occurring simultaneously. The temporal ordering among the events is induced by the absolute timestamps associated with the events [8]. Usually, all the transactions of a customer are together viewed as a sequence, called customer-sequence, where each transaction is represented as an item set in that sequence and all the transactions are listed in a certain order with regard to the transaction-time [4]. The process of mining sequential patterns from a customer transaction database is described as follows:

Let D be a set of customer transactions where each transaction T consists of a customer-id, a transaction time and a set of items involved in the transaction. Let $I = \{i_1, i_2, ..., i_m\}$ 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 $\langle s_1, s_2, ... s_n \rangle$, where s_j, $j \in 1, ..., n$, an item is set. A k-sequence is a sequence of k items (or of length k). A sequence $\langle s_1, s_2, ... s_n \rangle$ is a sub-sequence of another sequence $\langle s'_1, s'_2, ... s'_m \rangle$ if there exist integers $i_1 < i_2 < .....i_j ..... < i_n$ such that $s_1 \subseteq s'_{i1}, s_2 \subseteq s'_{i2}, ... s_n \subseteq s'_{in}$ [9]. The problem of mining sequential patterns is to find all sequences s such that supp(s) ≥ *minsup* for a database D, given a support threshold *minsup*.

The task of discovering all frequent sequences in large databases is quite challenging, since the search space is extremely large. For instance, with m attributes there are $O(m^k)$ potentially frequent sequences of length k [10]. The factors, which make sequential pattern mining a very difficult and time-consuming one, are as follows; first, the formation of a pattern is not limited to single items but item sets. Second, neither the number of item sets in a pattern nor the number of items in an item set is known a priori. Third, patterns could be formed by any permutation, of any combination of possible items in the database [11]. Since its introduction in 1995 by Agrawal *et al.* [7], the task of mining sequential patterns has received a great deal of attention. Following their work, there have been many studies on sequential pattern mining and its applications [14, 15]. 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.

2. LITERATURE REVIEW

The literature presents with a huge number of approaches for constraint-based sequential pattern mining and incremental mining of sequential patterns. In recent times, developing approaches for incremental mining of constraint-based sequential patterns has gained immense importance in real life applications. A concise review of some recent researches related to the incremental mining of sequential patterns is presented here.

Jiaxin Liu [17] have proposed a data storage structure, called frequent sequence tree, and give the construction algorithm of frequent sequence tree, called Con_FST. The root node of the frequent sequence tree stored the frequent sequence tree support threshold and the path from the root node to any leaf node represents a sequential pattern in the database. Frequent sequence tree stored all the sequential patterns with its support that meet the frequent sequence tree support threshold, so when the support was changed, the algorithm which uses frequent sequence tree as the storage structure could find all the sequential patterns without mining the database. Philippe Fournier *et al.* [20] have presented a Rule Growth, an algorithm for mining sequential rules common to several sequences. Unlike other algorithms, Rule Growth used a pattern-growth approach for discovering sequential rules such that it can be much more efficient and scalable. They have presented a comparison of Rule Growth's performance with current algorithms for three public datasets. The result showed that Rule Growth clearly outperforms current algorithms for all three datasets under low support and confidence threshold.

Jiaxin Liu [18] have proposed that the structure of sequence tree based on projected database, called sequence tree, and give the Steeps algorithm which was used to construct the sequence tree. Sequence tree was a data storage structure; it has been the similar in structure to the prefix tree. But, the sequence tree stores all the sequences in the original database. The path from the root node to any leaf node represents a sequence in the database. The structural characteristic of sequence tree makes it suitable for incremental sequential pattern mining. The result showed that the incremental mining algorithm of sequential patterns which uses the sequence tree as the storage structure for sequences outperformed Prefix Span in space cost on condition that the support threshold was smaller. To capture the dynamic nature of data addition and deletion,

Jen-Wei Huang *et al.* [16] have proposed a general model of sequential pattern mining with a progressive database while the data in the database may be static, inserted or deleted. In addition, they presented a progressive algorithm Pisa, standing for Progressive mIning of Sequential pAtterns, to progressively discover sequential patterns in defined time period of interest. The period of interest is a sliding window continuously advancing as the time goes by. Pisa utilizes a 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. The height of the sequential pattern tree proposed was bounded by the length of period of interest, thereby effectively limiting the memory space required by Pisa that is significantly smaller than the memory needed by alternative methods.

BTzung-Pei *et al.* [19] have proposed an incremental mining algorithm for maintaining sequential patterns based on the concept of pre-large sequences to reduce the need for rescanning original databases. Pre-large sequences were defined by a lower support threshold and an upper support threshold that act as gaps to avoid the movements of sequences directly from large to small and vice-versa. The algorithm does not require rescanning original databases until the accumulative amount of newly added customer sequences exceeds a safety bound, which depends on database size. Thus, as databases grow larger, the numbers of new transactions allowed before database rescanning was required also grow.

3. COMPARISION OF SEQUENTIAL PATTERN MINING ALGORITHMS

As described by Yen-Liang Chen and Ya-Han Hu [28] 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 concerns 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.

3.1 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 [27]:

(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

3.1.1. 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 ant monotonic (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: [27].

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

3.1.1.2 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.

3.1.1.3 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.

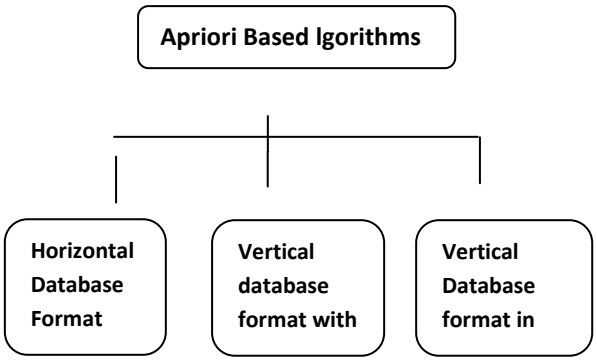

Fig. 1: Classification of Apriori based mining algorithm

3.1.1.4 GSP:

The GSP algorithm described by Agrawal and Shrikant [30] 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 pert 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.

3.1.1.5 SPIRIT:

The Novel idea of the SPIRIT algorithm is to use regular expressions as flexible constraint specification tool [21]. 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.

3.1.1.6 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 [36] 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

3.1.1.7 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 item set-extended child against min_sup recursively for efficient support-counting SPAM uses a vertical bitmap data structure representation of the database ,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. [24]

3.1.2. 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.

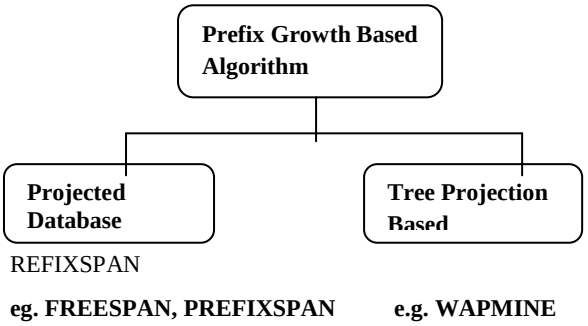

Fig 2: Classification of Prefix Growth based mining algorithm

3.1.2.1 FREESPAN:

FreeSpan [22] 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.

3.1.2.2 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.

3.1.2.3 PREFIXSPAN:

The PrefixSpan (Prefix-projected Sequential pattern mining) algorithm presented by Jian Pei, Jiawei Han and Helen Pinto [23] representing the pattern-growth methodology, which finds the frequent items after scanning the sequence database once. The database is projected 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.

3.2 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.

3.2.1 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 [26] 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. [29]

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. [29]

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 behaviour implied by a sequential pattern must occur in a reasonable period. [29] *Target-Oriented* A target-oriented sequential pattern is a sequential pattern with a concerned item set 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 item sets only, and thus, most sequential patterns discovered by using traditional algorithms are irrelevant and useless. [30]

3.2.2 .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 [25] 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. Attunes, A. L. Oliveira, [23] 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.

4. 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

4.1 Characteristics of Sequential Pattern Mining Algorithm are::

4.1.1 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.

*4.1.2 BFS-Based Approach Vs. DFS-Based Ap***proach** 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.

4.1.3 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 2l 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.

4.1.4 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.

4.1.5 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.

4.2 Experimental Analysis Done By Researchers [28]

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 item set in maximal sequence It is assumed that number of items is 10,000 and on average, a frequent sequential pattern consists of four transactions

4.2.1 Comparison of Memory Usage

From the comparison graph of fig. 3, 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 present, 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.

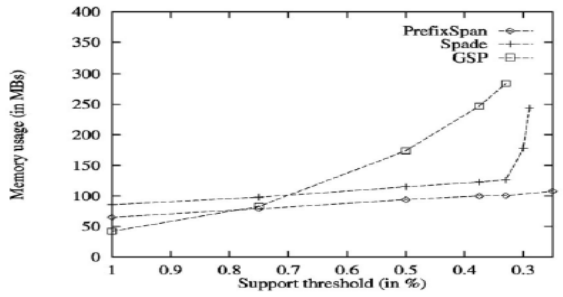

Fig. 3: Memory usage of algorithms on data set C200T5S10I2.5

4.2.2 Comparison of Time Complexity

As from the comparison graph from fig. 4, it is clear that both of the pattern growth algorithm, Freespan and Prefixspan are time efficient than Apriori based Algorithm.

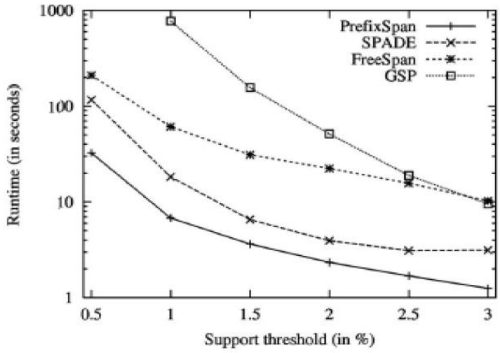

Fig.4: Performance of the four algorithms on data set C10T8S8I8

4.2.3 Comparison of Scalability

From the Experimental results shown in fig. 5, 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 pseudo projection tables on the other hand, both SPADE and GSP need memory space to hold candidate sequence patterns as well as the sequence databases

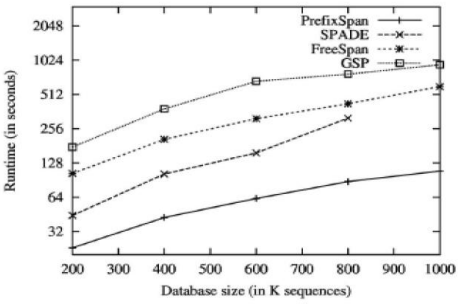

Fig.5: Scalability test of algorithms on data set T2:5S10I1:25, with min_support 0.5 present.

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:

4.2.4 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.

4.2.5 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.

4.2.6 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.

6. 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[31].

7. REFERENCES

[1] Sizu Hou, Xianfei Zhang, "Alarms Association Rules Based on Sequential Pattern Mining Algorithm," In proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 2, pp.556-560, Shandong, 2008.

[2] Jatin D Parmar and Sanjay Garg, "Modified Web Access Pattern (mWAP) Approach for Sequential Pattern Mining", Journal of computer Science, Vol. 6, No.2, pp.46-54, June 2007.

[3] Tarek Sobh, "Innovations and Advanced Techniques in Computer and Information Sciences", Springer, 2007, ISBN 978-1-4020-6268-1.

[4] Qiankun Zhao and Sourav S. Bhowmick, "Sequential Pattern Mining: A Survey" , Technical Report, CAIS, Nanyang Technological University, Singapore, No.118, 2003.

[5] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", Morgan Kaufman publishers, 2001, ISBN: 1-55860489-8.

[6] S. Myra, "Web usage mining for Web site evaluation", Communications of the ACM, vol. 43, No. 8, pp. 127–134, 2000.

[7] R. Agrawal and R. Srikant, "Mining Sequential Patterns", In Proceedings of the 11th International Conference on Data Engineering, pp. 3-14, Taipei, Taiwan, 1995.

[8] Salvatore Orlando, Raffaele Perego and Claudio Silvestri, "A new algorithm for gap constrained sequence mining", In Proceedings of the ACM Symposium on Applied Computing, pp. 540 – 547, Nicosia, Cyprus, 2004.

[9] Florent Masseglia, Pascal Poncelet and Maguelonne Teisseire, "Incremental mining of sequential patterns in large databases", Data & Knowledge Engineering, Vol. 46, No.1, pp. 97-121, 2003.

[10] M. Zaki, "Scalable data mining for rules", Technical Report Ph.D. Dissertation, University of Rochester, New York, 1998.

[11] Ming-Yen Lin and Suh-Yin Lee, "Interactive Sequence Discovery by Incremental Mining", An International Journal of Information Sciences-Informatics and Computer Science, vol. 165, no. 3-4 , pp.187 - 205, October 2004.

[12] M. J. Zaki, "Efficient enumeration of frequent sequences," Proceedings of the 7th International Conference on Information and Knowledge Management, Washington, USA, pp. 68-75, Nov.1998.

[13] Jian Pei, Jiawei Han and Wei Wang, "Constraint-based sequential pattern mining: the pattern-growth methods", Journal of Intelligent Information Systems, Vol:28, No: 2 ,pp:133-160, 2007.

[14] R. Srikant and R. Agrawal. "Mining sequential patterns: Generalizations and performance improvements". In Proc. of the 5th International Conference on Extending Database Technology (EDBT'96), pages 3–17, Avignon, France, September 1996.

[15] F. Masseglia, F. Cathalat, and P. Poncelet. "The PSP approach for mining sequential patterns". In Proc. of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery in Databases (PKDD'98), pages 176–184, Nantes, France, September 1998. Lecture Notes in Artificial Intelligence, Springer Verlag.

[16] Jen-Wei Huang, Chi-Yao Tseng, Jian-Chih Ou, Ming-Syan Chen, "A General Model for Sequential Pattern Mining with a Progressive Database," IEEE Transactions on Knowledge and Data Engineering, vol. 20, No. 9, pp. 1153-1167, 2008.

[17] Jiaxin Liu, "The design of storage structure for sequence in incremental sequential patterns mining," Networked Computing and Advanced Information Management (NCM), pp. 330 - 334, 2010.

[18] Jiaxin Liu, "The design of frequent sequence tree in incremental mining of sequential patterns," Software Engineering and Service Science (ICSESS), pp. 679-682, 2012.

[19] Tzung-Pei, Hong,Ching-Yao Wang and Shian-Shyong Tseng, "An Incremental Mining Algorithm for Maintaining Sequential Patterns Using Pre-large Sequences," Journal Expert Systems with Applications, Vol. 38, Issue 6,p p.7051-7058, 2011.

[20] Philippe Fournier,Viger,Roger Nkambou and Vincent Shin-Mu Tseng, "RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth," Symposium on Applied Computing, pp . 951-960, 2011.

[21] M. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential pattern mining with regular expression constraints", VLDB'99, 1999.

[22] 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.

[23] J. Pei, J. Han, B. Mortazavi-Asi, H. Pino, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth", ICDE'01, 2001.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Rong She ,Fei Chen,Ke Wang ,Martin Ester ,Jennifer L. Gardy , Fiona S. L. Brinkman, "Frequent-subsequence-based prediction of outer membrane proteins", Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining , pp: 436 - 445, 2003.

[31] Chetna Chand, Amir Thakkar, Amit Gamtra, International Journal of Soft Computing and Engineering",ISSN:2231-2307, Volume-2, Issue-1, March 2012

AUTHOR INFORMATION

V.Chandra Shekhar Rao obtained his Masters's degree in Computer Science and Engineering from JNT University of Hyderabad, Andhra Pradesh, India. Pursuing PhD in Computer Science & Engineering at JNTU Hyderabad, His research interests are in Knowledge Discovery, Data Mining, Databases. At present he is working as an Associate Professor in CSE dept. KITS, Warangal, Andhra Pradesh, INDIA.

Dr. P. Sammulal received PhD in Computer Science and Engineering from Osmania University in 2010. He received his B.E degree from Osmania University in 2000 and MTech degree from JNT University in Computer science and engg., in 2002. He has published about 18 papers in International/national conferences and International/national journals. His current research interests are distributed/parallel computing, Cluster computing, Grid computing, Network security and Data Mining. At present he is working as an Assistant Professor in JNTUH College of Engineering, Nachupally, karimnager (dist), INDIA.