International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 2, February 2013 ISSN 2319 - 4847

# A Comparative Study of Sequential Pattern Mining Algorithms

**Manan Parikh[1], Bharat Chaudhari[2] and Chetna Chand[3]**

[1, 2] MECSE, KITRC KALOL

[3]ASSISTANT PROFESSOR, KITRC KALOL

## ABSTRACT

***Sequential pattern mining is an important data mining problem with broad applications. However, it is also a difficult problem since the mining may have to generate or examine a combinatorial explosive number of intermediate subsequences. Most of the previously developed sequential pattern mining methods, such as GSP, explore a candidate generation-and-test approach [1] to reduce the number of candidates to be examined. However, this approach may not be efficient in mining large sequence databases having numerous patterns and/or long patterns. A comprehensive performance study shows that PrefixSpan, in most cases, outperforms the apriori-based algorithm GSP, FreeSpan[2], and SPADE [3] (a sequential pattern mining algorithm that adopts vertical data format), and PrefixSpan[4] integrated with pseudoprojection is the fastest among all the tested algorithms. Furthermore, this mining methodology can be extended to mining sequential patterns with user-specified constraints. The high promise of the pattern-growth approach may lead to its further extension toward efficient mining of other kinds of frequent patterns, such as frequent substructures.***

**Keywords:** Data mining algorithm, sequential pattern, frequent pattern, transaction database, sequence

## 1. INTRODUCTION

The sequential pattern mining problem was first introduced by Agrawal and Skrikant [5] and can be stated as follows. We are given a set of sequences, called data-sequences, as the input data. Each data-sequence is a list of transactions, where each transaction contains a set of literals, called items. Given a user-specified minimum support threshold, sequential pattern mining is to find all of the frequent subsequences in the sequence database, i.e., the subsequences whose ratios of appearance exceed the minimum support threshold. As an example, a retailer may find a sequential pattern “having bought a notebook, a customer returns to buy a PDA and a WLAN card” from its customer purchase database.

Many approaches ([6]) have been proposed to extract sequential patterns from sequence databases. Most of early works focus on the following two directions:

1. **Improve the efficiency of the mining process**. This direction of research focuses on the efficient mining of sequential patterns in time-related data ([7], [4], [8], [2],). In general, these methods can be categorized into three classes: (1) Apriori-based, horizontal formatting method, such as GSP [1]; (2) Apriori-based, vertical formatting method, such as SPADE [3]; (3) Projection-based pattern growth method, such as PrefixSpan([8]) and (4) Apriori-based candidate generation and pruning using depth-first traversal, such as SPAM [9].
2. **Extend the mining of sequential patterns to other time-related patterns**. In addition to improving the efficiency of the mining process, another important direction of research is to find other types of pattern in time-related databases. Important research in this category includes finding frequent episodes in event sequences ([10]); finding frequent traversal patterns in a web log ([11]); finding cyclic patterns in a time-stamped transaction database ([12]).

## 2. SEQUENTIAL PATTERN MINING ALGORITHMS

As per data mining theory, sequential pattern mining algorithm can be broadly divided into three groups: Aprioribased (GSP, SPADE, SPAM), patterngrowth (FreeSpan, PrefixSpan), early pruning (LAPIN-SPAM). There are lots of algorithms for sequential pattern mining but here I show some of good algorithms that I studied and advantages and disadvantages of those algorithms.

### 2.1 Apriori Algorithm and Its Extension to Sequence Mining

A sequence is a time-ordered list of objects, in which each object consists of an itemset, with an itemset consisting of all items that appear (or were bought) together in a transaction (or session). Note that the order of items in an itemset does not matter. A sequence database consists of tuples, where each tuple consists of a customer id, transaction id, and an

itemset. The purpose of sequence mining is to find frequent sequences that exceed a user-specified support threshold. Note that the support of a sequence is the percentage of tuples in the database that contain the sequence.
An example of a sequence is {(ABD), (CDA)}, where items ABD indicate the items that were purchased in the first transaction of a customer and CDA indicate the items that were purchased in a later transaction of the same customer.
According frequency of the item we make large item set and mapping of that item sets. Here mini_sup=2.

**Table 1** shows the transaction sequences of different customers

| Customer Id | Transaction Time | Iteam Bought |
|---|---|---|
| 1 | Oct 25 | 30 |
| 1 | Oct 30 | 90 |
| 2 | Oct 10 | 10,20 |
| 2 | Oct 15 | 30 |
| 2 | Oct 20 | 40,60,70 |
| 3 | Oct 25 | 30,50,70 |
| 4 | Oct 25 | 30 |
| 4 | Oct 30 | 40,70 |
| 4 | Oct 25 | 90 |
| 5 | Oct 12 | 90 |

**Table 2** Make customer sequence according to time duration

| Customer Id | Customer Sequence |
|---|---|
| 1 | <(30) (90)> |
| 2 | <(10 20) (30) (40 60 70)> |
| 3 | <(30 50 70)> |
| 4 | <(30) (40 70 ) (90)> |
| 5 | <(90)> |

**Table 3** Mapping of Iteamsets

| Large Iteam set | Mapped To |
|---|---|
| (30) | 1 |
| (40) | 2 |
| (70) | 3 |
| (40 70) | 4 |
| (90) | 5 |

**Applying basic Apriori Algorithm:**
The Apriori algorithm was first proposed by Agrawal in [5], for the discovery of frequent itemsets. It is the most widely used algorithm for the discovery of frequent itemsets and association rules. The main concepts of the Apriori algorithm are as follows:

1. Any subset of a frequent itemset is a frequent itemset.
2. The set of itemsets of size k will be called $C_k$.
3. The set of frequent itemsets that also satisfy the minimum support constraint is known as $L_k$. This is the seed set used for the next pass over the data.
4. $C_{k+1}$ is generated by joining $L_k$ with itself. The itemsets of each pass have one more element than the itemsets of the previous pass.
5. Similarly, $L_{k+1}$ is then generated by eliminating from $C_k$ those elements that do not satisfy the minimum support rule. Because the candidate sequences are generated by starting with the smaller sequences and progressively

increasing the sequence size, Apriori is called a breadthfirst approach. An example is shown below, where we require the minimum support to be two. The left column shows the transaction ID and the right column shows the items that were purchased

**Table 4** Support Counting of 1-Sequence

| L1 | |
|---|---|
| 1-sequence | Support |
| <1> | 4 |
| <2> | 2 |
| <3> | 4 |
| <4> | 4 |
| <5> | 4 |

**Table 5** Support Counting of 2-Sequence

| L2 | |
|---|---|
| 2-sequence | Support |
| <1 2> | 2 |
| <1 3> | 4 |
| <1 4> | 3 |
| <1 5> | 3 |
| <2 3> | 2 |
| <2 4> | 2 |
| <3 4> | 3 |
| <3 5> | 2 |
| <4 5> | 2 |

**Table 6** Support Counting of 3-Sequence

| L3 | |
|---|---|
| 3-sequence | Support |
| <1 2 3> | 2 |
| <1 2 4> | 2 |
| <1 3 4> | **3** |
| <1 3 5> | 2 |
| <2 3 4> | 2 |

**Table 7** Support Counting of 4-Sequence

| L4 | |
|---|---|
| 4-sequence | Support |
| <1 2 3 4> | 2 |

As noted in [13], the problem with the Apriori approach is that it can generate a very large number of candidate sequences. For example, to generate a frequent sequence of length 100, it must generate 2100 candidates, because these are its subsequences, which are also frequent.

**2.2 GSP (Generalized Sequential Pattern) mining algorithm.**

GSP was proposed by the same team as the Apriori algorithm in [18], and it can be 20 times faster than the Apriori algorithm because it has a more intelligent selection of candidates for each step and introduces time constraints in the search space.

Agrawal and Srikant formulated GSP to address three cases:

1. The presence of time constraints that specify a maximum time between adjacent elements in a pattern. The time window can apply to items that are bought in the same transaction or in different transactions.
2. The relaxation of the restriction that the items in an element of a sequential pattern must come from the same transaction.
3. Allowing elements in a sequence to consist of items that belong to taxonomy.

The algorithm has very good scalability as the size of the data increases.

The GSP algorithm is similar to the Apriori algorithm. It makes multiple passes over the data as is shown below:

- In the first pass, it finds the frequent sequences.In other words, it finds the sequences that have minimum support. This becomes the next seed set for the next iteration.
- At each next iteration, each candidate sequence has one more item than the seed sequence.

Two key innovations in the GSP algorithm are how candidates are generated and how candidates are counted. Regarding candidate counting, a hash table is used. Regarding candidate generation, a novel definition of a contiguous subsequence is used, which is described below.

The above algorithm looks like the Apriori algorithm. One main difference is however the generation of candidate sets. Let us assume that:

$A \rightarrow B$ and $A \rightarrow C$ are two frequent 2-sequences. The items involved in these sequences are (A, B) and (A, C) respectively. The candidate generation in a usual Apriori style would give (A, B, C) as a 3-itemset, but in the present context we get the following 3-sequences as a result of joining the above 2- sequences

$A \rightarrow B \rightarrow C$, $A \rightarrow C \rightarrow B$ and $A \rightarrow BC$

The candidate–generation phase takes this into account.

However, as the authors in [4] note, there are three inherent costs associated with GSP:

- When dealing with large sequence databases, a very large number of candidate sequences could be generated.
- f databases are required.
- When dealing with long sequences, the number of candidates could experience exponential growth. This could be the case in DNA sequences or stock market data.

**2.3 SPADE (Sequential PAttern Discovery using Equivalent Class).**

In [3], Zaki proposes an algorithm that takes a different approach than Apriori and GSP for the discovery of sequential patterns. Instead of assuming a horizontal database, where a customer has a set of transactions associated with him, SPADE assumes a vertical database, where each item is associated with a customer id and a transaction id. The support of each k-sequence is determined by the temporal join of k-1 sequences that share a common suffix [14].

As we saw, both Apriori and GSP make multiple database scans. GSP also uses a hash structure. The innovation of the SPADE algorithm is that it decomposes the original problem into smaller problems and uses lattice search techniques, to solve the problems independently in memory. The important contribution of this algorithm is that all sequences are discovered in only three database scans. This becomes even more important if one considers how large the search space is demanding applications. In [16], Zaki et al. propose an algorithm that finds frequent sequences that always precede plan failures, with applications in emergency situations. The algorithm (PlanMine) is based on the SPADE algorithm and it successively filters out uninteresting sequences. In the first step, sequences that are not related to plan failures are filtered out. In the second step, it filters out frequent sequences that appear frequently in plan failures, but also have high support in plans that did not fail. In the third step, redundant patterns are removed, which are patterns that have subpatterns in both good and bad plans. In the final step, dominated patterns are removed. These are patterns that have subsequences that have lower support in good plans and higher support in bad plans than the original sequence in which they are contained.

**2.4 PrefixSpan**

In [4], the authors propose a technique for sequential patterns mining, PrefixSpan, that deviates from the standard ways of generating candidates and testing them, such as GSP and SPADE. Instead, they propose a method that has the following two key features:

- It is projection-based. What is projected is the actual database. Specifically, the sequence database is iteratively projected into smaller projected databases. Han, Pei, et al. first developed the FreeSpan algorithm [2], using projected databases.
- It grows patterns sequentially in the projected databases, by investigating only locally frequent segments. Specifically, in a database scan, the frequent patterns above a certain support are recorded. Then for each one of the

frequent patterns, their ending locations are used as the beginning locations of new scans. Frequent patterns discovered in this scan are then appended to the frequent patterns discovered in the previous scan. As noted in [17], each projected database has the same prefix subsequence.

One of the computational costs of PrefixSpan is the creation of the projected databases. The authors propose pseudoprojection as a way to reduce the number and size of the projected databases. In this method, the index of the sequence and the starting position of the projected suffix are recorded, instead of performing a physical projection.

In experimental results on different data sets, PrefixSpan outperformed GSP and SPADE. In a comprehensive performance study, the authors explain the reasons PrefixSpan outperformed the other methods:

- •It grows patterns without candidate generation.
- •The projections are an effective way to perform data reduction.
- •The memory space utilization is approximately steady.

In [18], a variation of PrefixScan, called CloSpan, is proposed that is tailored to detect closed subsequences only. A closed sequential pattern is one for which there is no superset sequence (with the same support) containing the pattern.

### 2.5 The SPAM and I-SPAM Algorithms

In [9], the authors propose a sequential pattern mining technique that utilizes a bitmap representation called SPAM. The algorithm is the first sequential mining method that utilizes a depth-first approach to explore the search space. Combining this search strategy with an effective pruning technique that reduces the number of candidates makes the algorithm particularly suitable for very long sequential patterns. However, the algorithm requires that the whole database can be stored in main memory, which is the main drawback of the algorithm.

As sequences are generated traversing the tree, two types of children are generated from each node: sequence-extended sequences (sequenceextension step or S-step) and itemset-extended sequences (item-extension step or I-step). Finally, an efficient representation of the data is used, which is a vertical bitmap representation. The bitmap is created for each item as follows: If item A belongs to transaction j, then a 1 is recorded for A in transaction j. Otherwise a 0 is recorded.

SPAM was compared against SPADE and PrefixSpan using several small, medium, and large data sets. On small data sets, SPAM was about 2.5 times faster than SPADE, but PrefixSpan was a bit faster than SPAM on very small data sets. On large data sets, SPAM was faster than PrefixSpan by about an order of magnitude. The authors attributed the fast performance on large data sets to its efficient bitmap-based representation.

An improved version of SPAM is proposed in [17], called I-SPAM.

In this version, approximately half of the maximum memory requirement of SPAM is needed with no sacrifice in the performance time.

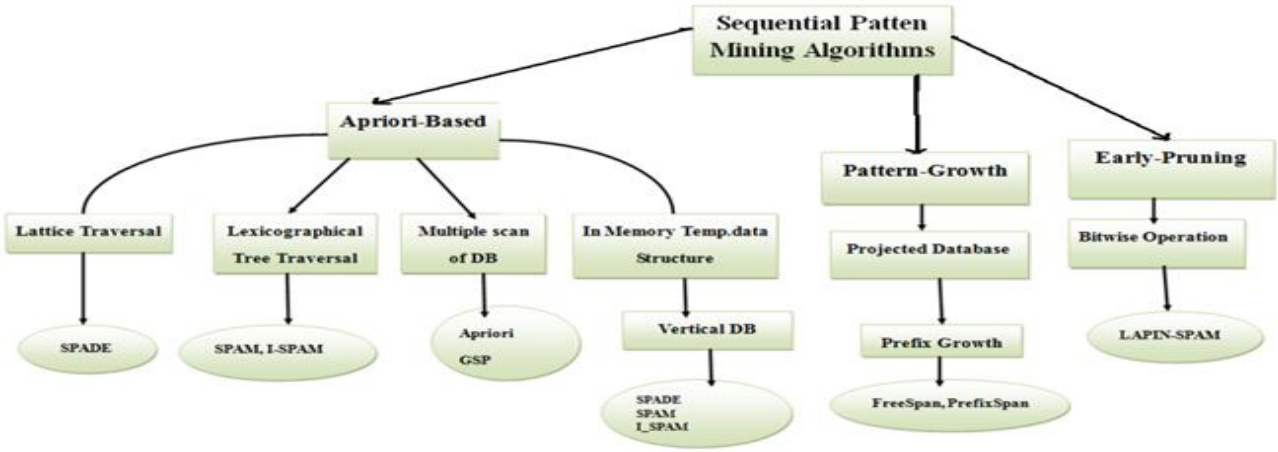


**Figure 1** Sequential Pattern Mining Algorithms.

### 3. Taxonomy of existing sequential pattern-mining algorithms

Taxonomy of existing sequential pattern-mining algorithms is provided in Figure 3.2 in this section, which lists the algorithms, showing a comparative analysis of their different important features. This proposed taxonomy is composed of three main categories of sequential pattern-mining algorithms, namely, apriori-based, pattern-growth and early-pruning algorithms.

Algorithms mainly differ in two ways:

1. The way in which candidate sequences are generated and stored. The main goal here is to minimize the number of candidatesequences 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.

| | Apriori-Based | | Pattern Growth | | | | | | Early- Puring |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | Generate & Test | Multiple scan of database | Sampling and/or compression | Candidate sequence pruning | Search space partitioning | Depth first traversal | Prefix Growth | Memory Only | Vertical Projection of DB |
| Apriori | X | X | X | | X | | | | |
| GSP | X | X | | X | | | | | |
| SPADE | X | | | X | X | X | | | X |
| FreeSpan | | | | | X | | | X | |
| PrefixSpan | | | | X | X | | X | X | |
| SPAM | X | | | | | X | | X | X |
| I-SPAM | X | | | | | X | | X | |

**Figure 2** Tabular Taxonomy of Sequential Pattern Mining Algorithms. [19]

Careful investigation of Figure 2 shows how slow the Apriori-based SPAM algorithm could become as data set size grows from medium (| D |=200K) to large (| D |=800K), due to the increased number of AND operations and the traversal of the large lexicographical tree; although it is a little faster than PrefixSpan on large data sets due to the utilization of bitmaps as compared to the projected databases of PrefixSpan.

**Table 1** Comparative analysis of algorithm performance. [19] The symbol "-" means an algorithm crashes with the parameters provided, and memory usage could not be measured.

| Algorithm | Data Set size | Minimum Support | Execution time(sec) | Memory Usage(MB) |
|---|---|---|---|---|
| GSP Apriori Based | Medium (\|D\|=200k) | Low (0.1%) | >3600 | 800 |
| | | Medium (1%) | 2126 | 687 |
| | Large (\|D\|=800k) | Low (0.1%) | - | - |
| | | Medium (1%) | - | - |
| SPAM Apriori Based | Medium (\|D\|=200k) | Low (0.1%) | - | - |
| | | Medium (1%) | 136 | 574 |
| | Large (\|D\|=800k) | Low (0.1%) | - | - |
| | | Medium (1%) | 674 | 1052 |
| PrefixSpan Pattern Growth | Medium (\|D\|=200k) | Low (0.1%) | 31 | 13 |
| | | Medium (1%) | 5 | 10 |
| | Large (\|D\|=800k) | Low (0.1%) | 1958 | 525 |
| | | Medium (1%) | 798 | 320 |

## 4. CONCLUSIONS AND FUTURE WORK

This Paper presents a taxonomy of sequential pattern-mining algorithms, and shows that current algorithms in the area can be classified into three main categories, namely, apriori-based, pattern-growth, and early-pruning with a fourth category as a hybrid of the main three. A thorough discussion of 13 characteristic features of the four categories of

algorithms, with an investigation of the different methods and techniques, is presented. This investigation of sequential pattern-mining algorithms in the literature shows that the important heuristics employed include the following: using optimally sized data structure representations of the sequence database; early pruning of candidate sequences; mechanisms to reduce support counting; and maintaining a narrow search space.

## REFREMCES

[1]Agrawal, R. and R. Srikant, Mining Sequential Patterns: Generalizations and Performance Improvements, Lecture Notes in Computer Science— Advances in Database Technology, pp. 3–17, 1996.
[2]J. Pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. C. Hsu, FreeSpan: frequent pattern-projected sequential pattern mining, Proceedings of 2000 International Conference on Knowledge Discovery and Data Mining, (2000), pp. 355-359.
[3]Zaki, M., SPADE: An Efficient Algorithm for Mining Frequent Sequences, Machine Learning, vol. 42, no. 1/2, pp. 31–60, and 2001.
[4]Pei, J. et al., Mining Sequential Patterns by Pattern Growth: The PrefixSpan Approach, IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, pp. 1424–1440, Nov. 2004.
[5]Agrawal, R, T. Imielinski, and A. Swami, Mining Association Rules Between Sets of Items in Large Databases, Proc. ACM SIGMOD nternational Conference Management of Data, vol. 22, pp. 207–216, 1993.
[6]M. S. Chen, J. Han, P. S. Yu, Data mining: an overview from a database perspective, IEEE Transactions on Knowledge and Data Engineering, Vol. 8, Iss. 6, (1996), pp. 866-883.
[7]R. Agrawal, C. Faloutsos, & A. Swami. Efficient similarity search in sequence databases. Proceedings of Conference on Foundations of Data Organization and Algorithms, (1993), pp. 69-84.
[8]J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, M. C. Hsu, PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. Proceedings of 2001 International Conference on Data Engineering, (2001), pp. 215-224.
[9]Ayres, J. et al., Sequential Pattern Mining Using a Bitmap Representation, Proceedings of Conference on Knowledge Discovery and Data Mining, pp. 429–435, 2002.
[10]H. Mannila, H. Toivonen, A. I. Verkamo, Discovery of frequent episodes in event sequences, Data Mining and Knowledge Discovery, Vol. 1, Iss. 3, (1997), pp. 259-289.
[11]R. Srikant, Y. Yang, Mining web logs to improve website organization, Proceedings of the Tenth International World Wide Web Conference, Hong Kong, (2001).
[12]Y. L. Chen, M. C. Chiang, M. T. Kao, Discovering time-interval sequential patterns in sequence databases, Expert Systems with Applications, Vol. 25, Iss. 3, (2003), pp. 343-354.
[13]Agrawal, R. and R. Srikant, Mining Sequential Patterns, Proceedings of the 11th Int. Conference on Data Engineering, pp. 3–14, March 1995.
[14]Abramowicz, W. and J. Zurada, Knowledge Discovery for Business Information Systems, Springer, 2001.
[15].Masseglia, F., F. Cathala, and P. Poncelet, The PSP Approach for Mining Sequential Patterns, Lecture Notes in Computer Science, Principles of Data Mining and Knowledge Discovery, vol. 1510, pp. 176–184, 1998.
[16]Zaki, M., N. Lesh, and M. Ogihara, PlanMine: Sequence Mining for Plan Failures, Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 369–373, 1998.
[17]Wu, C.L., J.L. Koh, and P.Y. An, Improved Sequential Pattern Mining Using an Extended Bitmap Representation, Lecture Notes in Computer Science, vol. 3588, pp. 776–785, 2005.
[18]Yan, X., J. Han, and R. Afshar, CloSpan: Mining Closed Sequential Patterns in Large Datasets, Proceedings of 2003 SIAM nternational Conference Data Mining, 2003.
[19]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.