A survey on improving the efficiency of prefix span sequential pattern mining algorithm

K Suneetha  
Sr. Lecturer,  
SVEC,  
Tirupati, AP, India.  
umasuni.k@gmail.com

Dr. M Usha Rani  
Associate Professor,  
Dept. of Computer Science, SVVMV,  
Tirupati, AP, India.  
musha_rohan@yahoo.com

Abstract—Sequential pattern mining is an important data mining problem with broad applications. Over the last decade many algorithms have been introduced. In this paper, a systematic survey of some of sequential pattern mining algorithms is performed by classifying into various categories. Most of the previously developed sequential pattern mining algorithms such as GSP, SPIRIT, SPADE uses apriori-based approach by exploring a candidate generation-and-test approach [8], to reduce the number of candidates examination. Then a comprehensive study on Prefixspan algorithm, a projection-based sequential pattern-growth approach for efficient mining of sequential patterns is performed which shows that PrefixSpan, in most cases, outperforms the apriori-based algorithm GSP, FreeSpan, and SPADE [7]. Later to further improve mining efficiency, the traditional sequential pattern mining algorithm PrefixSpan is modified by considering sufficient data structure Seq-Tree framework and separator database to reduce the execution time and memory usage. Also by considering different constraints such as monetary and compactness, enables to discover more user-specific patterns which are valuable and profitable.

Keywords- Apriori-based approach, Pattern growth approach, Frequent Patterns, Projection database, Sequence database.

I. INTRODUCTION

One of the real universal challenges is to find actionable knowledge from large amounts of data. In this study we focus on mining sequential patterns effectively. Sequential Pattern Mining was first introduced by Agrawal and Srikant[9]. From a given set of sequences, sequential pattern mining algorithms finds complete set of frequent subsequences that satisfy minimum support threshold value which are hidden in a database. These can be highly efficient, scalable, involving only a small number of database scans. Sequential pattern mining is an important data mining problem with broad applications including customer purchase behavior analysis, web-log analysis, medical treatments, natural disasters, science and engineering processes, stocks and markets, DNA sequences and gene structures.

A sequence database consists of ordered elements or events. A sequence is said to be sequential pattern if its support count is greater than or equal to min-sup value. One of the main problem in sequential pattern mining is to find the complete set of sequential patterns with respect to a given sequence database and a support threshold min_sup.

II. APPROACHES OF SEQUENTIAL PATTERN MINING

Recently many approaches in sequential pattern mining have been proposed. The earliest and the most influential methods for sequential pattern mining are classified into 2 approaches. They are 1) Apriori based approach and 2) Pattern growth approach. Algorithms using Apriori-based approaches are SPADE which is a Lattice Traversal, SPAM, BIDE, HVSM are Lexicographical tree traversal. Whereas GSP uses Horizontal database format.

Pattern growth approach is categorized into Tree projection and Projected database. Under Tree projection we have Prefix growth (PSP, PLWAP) and suffix growth (WAP-Mine, FS-Miner). Under projected database we have FREESPAN, PREFIXSPAN, SPARSE (Prefix growth) and LAPIN-Suffix (Suffix-growth).

A. APRIORI-BASED MINING ALGORITHMS:

The Apriori (Agrawal and Srikant 94) and AprioriAll (Agrawal and Srikanth 95) are the basis for many number of algorithms that depend greatly on the apriori property which states the fact that if a sequence S is not frequent, then none of the super-sequences of S can be frequent. The main idea is to iteratively generate the set of candidate patterns of length (k+1) from the set of frequent-patterns of length k (for k ≥ 1), and check their corresponding occurrence frequencies in the database. It achieves good performance by reducing the size of candidate sets.

a) GSP (Generalized Sequential Pattern Mining): The GSP algorithm [10] is based on apriori principle which is described by Agrawal and Shrikant[1996] mines sequential patterns by using a candidate subsequence generation-and-test approach. It makes multiple passes over the data. At 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, 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.

By studying the performance of GSP with AprioriAll algorithm, it is observed that GSP is much faster than AprioriAll at low levels of minimum support. The following are the reasons of why GSP is better than AprioriAll. 1. GSP counts fewer candidates than AprioriAll. 2. AprioriAll has to first find which frequent itemsets are present in each element of a data-sequence during the data transformation, and then find which candidate sequences are present in it. This is typically somewhat slower than directly finding the candidate sequences.

The bottlenecks of GSP are: 1. A huge set of candidate sequences are generated. 2. Multiple scans of database are needed and 3. It is inefficient for mining long sequential patterns.

b) SPIRIT (Sequential Pattern Mining with Regular Expression Constraints): SPIRIT (Garofalakis et al., 1999) is a family of algorithms for sequential pattern mining with regular expression constraints. SPIRIT algorithm uses regular expressions [6] as flexible constraint specification tool. On the mined patterns a generic user-specified constraint is involved which is considerably versatile and powerful restrictions are forced inside the mining process. But this algorithm uses less restrictive and more relaxed version of constraint. Many algorithms are available but each will differ in the degree to which the constraints are used to prune the search space while discovering patterns. A constraint specification tool called Regular Expressions (REs) uses two important factors. 1) REs provide a simple, natural syntax for the succinct specification of families of sequential patterns. 2) REs possess sufficient expressive power for specifying a wide range of interesting, non-trivial pattern constraints.

c) SPADE (Sequential Pattern Discovery using Equivalent Class): This algorithm was proposed by Zaki[2001]. This is a vertical format sequential pattern mining method. SPADE first maps the sequence database to a vertical id-list database format which is a large set of items <SID (Sequence ID), EID (Event ID)>. Sequential pattern mining is performed by growing the subsequences (patterns) one item at a time by Apriori candidate generation.

According to [7] the performance gap between SPADE and GSP algorithms increases with decreasing minimum support. SPADE is about twice as fast as GSP at lower values of support. SPADE outperforms GSP as:

1. SPADE uses only simple temporal join operation on id-lists. As the length of a frequent sequence increases, the size of its id-list decreases, resulting in very fast joins.

2. It doesn’t use complex data structure and so there isn’t any overhead when generating and searching through them.

3. SPADE successively restricts the search space needing only three database scans thus reducing the I/O costs.

But it suffers with additional computation time in order 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.

Apriori-based algorithms usually use a candidate “generate-and-test” type of approach, generate huge no of candidate sets and the search is not focusing on a restricted portion of initial database. It also suffers from the following two nontrivial costs: 1) It is costly to handle a huge number of candidate sets. 2) It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns.

So to avoid candidate generation-and-test and utilize some novel data structures to reduce the cost in frequent-pattern mining pattern growth approach which have been proposed in the early 2000’s. Its main idea is to follow a divide and conquer methodology and mine the sequential patterns without candidate generation and to focus the search on a restricted portion of initial database.

B. PATTERN GROWTH BASED MINING ALGORITHMS:

A frequent pattern growth method called FP-growth [5] has been developed for efficient mining of frequent patterns without candidate generation. The method uses a data structure called FP-tree. 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.

a) FRESESPN (Frequent pattern projected Sequential pattern mining): To reduce the cost of scanning multiple projected databases FreeSpan [3] was developed. It uses divide-and-conquer approach to recursively project the sequence database into projected databases while growing subsequence fragments in each projected database. This process partitions both the data and the set of frequent patterns to be tested, and confines each test being conducted to the corresponding smaller projected database. FreeSpan first scans the database, collects the support for each item, and finds the set of frequent items. Frequent items are listed in support descending order. Each projection partitions the database and confines further testing to progressively smaller and more manageable units.

According to the results of [3] Freespan outperforms GSP as follows:

1. Freespan projects a large sequence database recursively into a small set of projected sequence database based on currently mined frequent sets
2. Freespan reduces the cost of scanning multiple projected databases
3. It achieves better performance for large set of sequential patterns
But the trade-off is a considerable amount of sequence duplication as the same sequence could appear in more than one projected database.

b) PREFIXSPAN: Prefixspan is developed by Jian Pei, Jiawei Han and Wei Wang [4] based on the idea of database projection and sequential pattern growth. This algorithm examines only the prefix subsequences after scanning the sequence database once and then projects their corresponding postfix subsequences into projected database likewise sequential pattern are grown in each projected database by exploring only local frequent sequences.

Prefixspan algorithm doesn’t generate and tests candidate sequences, non-existent in a projection database. Projected database keeps on shrinking because only the suffix subsequences of a frequent prefix are projected into a projected database.

The key advantages of PrefixSpan are:
1) It does not generate any candidates.
2) It only counts the frequency of local items.
3) It utilizes a divide-and-conquer framework
4) Its performance is much better than both GSP and Freespan.

But the major cost of PrefixSpan is the construction of projected databases. To further improve mining efficiency, bi-level projection and pseudo-projection can be used.

To further improve mining efficiency, pseudo and bi-level projections can be used. Pseudo-projection is used when database can be held in the main memory. However, it is not efficient if the sequential database cannot be held in main memory.

A bi-level projection method is used, which projects databases not at every level but at every two levels. By comparing level-by-level with bi-level projection, bi-level projection reduces the cost of database projection and leads to better performance when the database is large and with low support threshold. The following table provides the comparative study of various sequential pattern mining algorithms.

### III. IMPROVEMENTS OF PREFIX PLAN ALGORITHM

A) I-PREFIXSPAN: An improved version of PrefixSpan named I-PrefixSpan [2] proposed by Dhany Saputra [2007]. The general idea of I-PrefixSpan is to use sufficient data structure for Seq-Tree framework and separator database to reduce the execution time and memory usage. It improves PrefixSpan in two ways: 1) by implementing ample data structure Seq-Tree framework for in-memory database sequence and constructs index set, and 2) a separator database to store the transaction alteration signs.

Seq-Tree is a general tree with two characteristics: 1) all leaves must be located at the same depth and 2) the height of the tree is at least 2. The ArrayIntList is used for in-memory.

<table>
<thead>
<tr>
<th>GSP</th>
<th>SPIRIT</th>
<th>SPADE</th>
<th>FREESPN</th>
<th>PREFIXSPAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apriori based</td>
<td>Apriori based</td>
<td>Apriori based</td>
<td>Pattern Growth based</td>
<td>Pattern Growth based</td>
</tr>
<tr>
<td>Uses Candidate generation and test approach</td>
<td>Uses Regular Expressions (REs) as a flexible Constraint</td>
<td>Uses vertical format sequential pattern mining method</td>
<td>Uses divide-and-conquer approach</td>
<td>Uses Projected database concept</td>
</tr>
<tr>
<td>Requires Multiple database scan.</td>
<td>Requires Multiple scans</td>
<td>Requires only three database scans</td>
<td>Reduces the cost of scanning multiple projected databases</td>
<td>Requires single database scan</td>
</tr>
<tr>
<td>Generates long sequential pattern, large number of candidates</td>
<td>Generates long sequential pattern that satisfies user-specified RE constraints</td>
<td>Generates large number of patterns, many of them are trivial or useless</td>
<td>Projects a large sequence database recursively into a set of small projected sequence databases</td>
<td>Generates long sequential pattern and less number of projected databases</td>
</tr>
<tr>
<td>Generates some candidates which doesn’t have any existence in sequence database</td>
<td>Generates fewer candidates which have the potential to be frequent for higher values of minimum support. Using equivalence classes on frequent sequences, the original problem decomposes into smaller sub-problems</td>
<td>Recursively project a sequence database into a set of smaller databases based on the current set of frequent patterns</td>
<td>Never generates any prefix which is not present in sequence database</td>
<td></td>
</tr>
<tr>
<td>Not good for those applications where low support thresholds are used</td>
<td>Performs well, even if it contains a large number of cycles of moderate length</td>
<td>Good for fast mining of sequential patterns in large databases</td>
<td>Good for large set of sequential patterns</td>
<td>Good for those applications where low support thresholds are used</td>
</tr>
<tr>
<td>Performance is poor than PrefixSpan algorithm</td>
<td>Performance is poor than GSP and poor than prefixspan</td>
<td>Performance is better than Prefixspan</td>
<td>Performance is better than GSP and poor than prefixspan</td>
<td>Performance is better than GSP algorithm</td>
</tr>
</tbody>
</table>
sequence database to store items. The ArrayIntList stores the offset for Index set. I-PrefixSpan uses separator database to find sequential patterns by comparing the index set of current pattern and the index set of items to be assembled or appended. A separator database helps to reduce the memory space and there is no need to traverse along all items inside all data sequences.

For a Dataset-1 (C1k|N150|T2|S2|t6), which contains 1000 sequences and 150 distinct items. Both the average number of items in a transaction and the average number of transaction in a sequence are set to 2. On average, a frequent sequential pattern consists of four transactions, and Fig. 1 below shows the processing time of PrefixSpan and I-PrefixSpan. I-PrefixSpan persistently outperforms PrefixSpan. The lower the minimum support, the clearer the excellence performance of I-PrefixSpan.

When the minimum support is 1%, I-PrefixSpan (7.67 seconds) is almost 4 times faster than PrefixSpan (30.269 seconds). When the minimum support is dwindled to 0.4%, I-PrefixSpan (46.853 seconds) is approximately 4.8 times faster than PrefixSpan (225.302 seconds). Moreover, when the support threshold is 0.2%, I-PrefixSpan (264.98 seconds) runs almost two orders of magnitude faster than PrefixSpan (19,582.97 seconds).

![Fig. 1 CPU performance of the two algorithms on Dataset-1](image)

**B) CFM-PREFIXSPAN:** It is tailored from traditional prefixspan algorithm by using pattern growth methodology [5]. It incorporates two constraints namely monetary and compactness which enables to discover more user-specific patterns. The constraint Monetary identifies sequential patterns that occur frequently in sequential database to find the significance of user buying sequences. Compactness identifies sequential patterns that occur within a reasonable time span.

This algorithm [1] first mines 1-length compact frequent patterns (1-CF) by using compactness threshold and support threshold. From this mined 1-CF patterns, 1-length compact frequent monetary (1-CFM) sequential patterns can be identified using monetary constraint. The mined 1-CF patterns are used to construct the projected database that is the collection of postfixes of sequence with regard to 1-CF pattern. By scanning the projected database once, we mine a set of 2-length compact frequent patterns. Then apply the monetary constraint on 2-length compact frequent patterns to get 2-CFM patterns. Likewise this process is repeated recursively until all CFM patterns are mined.

![Fig 2: Comparison graph of min-sup=1000 and T_m=10](image)

**REFERENCES**

1. B. Mallick, D. Garg & P.S. Grover, CFM-Prefixspan: A Pattern Growth Algorithm Incorporating Compactness and Monetary, ICIC International, Volume 8, Number 7(A), 2012 (pp 4509-4520)


