An Efficient Approach for Mining Sequential Pattern

Nidhi Pant, Surya Kant, Bhaskar Pant and Shashi Kumar Sharma

Abstract There are many different types of data mining tasks such as association rule mining (ARM), clustering, classification, and sequential pattern mining. Sequential pattern mining (SPM) is a data mining topic which is concerned with finding relevant patterns between data where values are delivered in sequence. Many algorithms have been proposed such as GSP and SPADE which work on Apriori property of generating candidates. This paper proposes a new technique which is quite simple, as it does not generate any candidate sets and requires only single database scan.

Keywords Sequential pattern mining · Sequence database

1 Introduction

Data mining refers to discovering of interesting patterns from large databases. Sequential pattern mining (SPM) process finds sets of data items that are present together frequently in some sequences of sequential database. In real world, huge amount of data continuously being collected and stored, many industries are becoming interested in mining sequential patterns from these databases.

SPM is a process of extracting those patterns, whose support exceeds predefined minimum support, i.e., it helps to extract patterns which reflect most frequent behavior in sequence database. To reduce very large number of sequence into most interesting sequential pattern, so as to fulfill the different user requirements, minimum support is introduced which prunes the sequential patterns with no interest.

Nidhi Pant · S.K. Sharma
Graphic Era Hill University, Dehradun, India

Surya Kant (✉)
IIT Roorkee, Roorkee, India
e-mail: suryak111@gmail.com

Bhaskar Pant
Graphic Era University, Dehradun, India

SPM is used in several domains such as in business organization to study customer behavior and also in area of web mining.

Many SPM algorithms have been proposed and prior algorithms among them are based on property of Apriori algorithms proposed by [1]. The property states that frequent pattern containing sub-pattern are also frequent. Based on this assumption many algorithms were proposed. Additionally, the Apriori property-based horizontal formatting method, generalized sequential pattern (GSP), has been presented in 1996 by same authors [2]. GSP is the exact algorithm to find frequent sequence in original database according to user-defined minimum support. It has the property that all subsequence of a frequent sequence must also be frequent and the property is as anti-monotone property. GSP works according to Apriori downward principle, where the candidate sequence length is generated from shorter length subsequence and thereafter candidates who have infrequent subsequence are pruned.

GSP makes multiple databases scan. In first scan, items of 1-sequence are recognized, eg: a_1, $a_2 \ldots a_m$. From recognized frequent items, candidate 2-sequence sets are generated from already identified first-level frequent sequences excluding rare subsequences from the first level. The algorithm consists of two important steps: 1. *Candidate generation*—Suppose a frequent set of Fk-1 sequence is given, performing a join of Fk-1 with itself will generate candidate sets for next scan and then pruning of database is done that eliminates sequences whose subsequences are not in frequent order. 2. *Support counting*—The search based on heap is used for efficient support counting. Finally, non-minimal sequences are removed.

Apriori property-based vertical formatting method (SPADE) was presented [3]. SPADE utilizes an equivalence class that decomposes initial problem into small subproblem, so that it can independently be solved in memory, using simple joins. In very first step spade generates 1-sequence items, items with single element, and this is done with one database scan. Second step, which consists of counting of 2-sequence, is performed by changing vertical database representation to horizontal database representation in the memory. It uses vertical representation of database; each row contains events uniquely identified by sequence id and event id. All sequences are recognized in three database scans. Many more algorithms like PrefixSpan, FreeSpan, and Sequential pattern mining using Bitmap Representation were also proposed for mining sequential patterns.

2 Literature Survey

The Apriori algorithm [1] has set a basis for those algorithms that largely depend on property of apriori and uses its Apriori-generate method. Apriori property says that all non-empty subsets of frequent item set should also be frequent and this property is called anti-monotonic property. Algorithms having priori property have disadvantage of maintaining frequency of each and every subsequence. To overcome from this problem, algorithms have discovered a way of calculating frequency and pruning sequences without maintaining count in each iteration.

A sequence database, where every single sequence is a set of transactions, is ordered by transaction time. The problem was analyzed by algorithm [2]. The problem of sequential mining is to identify all those patterns having support equal to more than a predefined minimum support and presented GSP algorithm. Algorithm [4] proposed a PSP algorithm. This algorithm uses a prefix tree that holds candidate sequence for each sequence along with their support count. When support threshold is very low algorithm performs very inefficiently.

Algorithm [3] introduced SPADE algorithm discovered for fast generation of sequential patterns. All existing algorithm of sequential mining makes multiple scans of database but SPADE discovers all frequent patterns in just three scans of database.

FreeSpan algorithm [5] uses projected databases for generation of database annotation that helps to quickly find frequent patterns. Shrinking factor of projected database is less than PrefixSpan.

In PrefixSpan algorithm [6], during scan of original database projected database is created. Ordering of lexicographic is maintained and is important as well. A lot of memory is occupied to store projections, especially when use of recursion is there.

Algorithm [7] introduces, generates, and tests feature, which are efficient for mining of sequences. Memory consumption is decreased to a magnitude order in this case.

Another technique [8] uses a rough set theory technique for finding local pattern from sequences, which is efficient for mining sequential patterns. Then [9] proposes the sequential pattern mining method specialized for analysis of learning histories of programming learning. [10] proposed a technique for identifying temporal relationships between medications and accurately predicting the next medication likely to be prescribed for a patient.

3 Proposed Algorithm

Given sequential database, scan the database ones, take the first sequence id, and from that generate all possible combinations with the help of following algorithm:

- Scan the database D.
- Set $i = 1$;
- For $i = 1$ to n;
- Select sequence Si
- Generate all possible combinations.
- Store them in tree.

GSP and SPADE algorithms of sequential pattern mining make multiple database scans and generate candidate sequences, which takes lot of time. But this new technique is far better as our technique makes only one database scan and also it does not generate candidate sequences. This proposed technique work on simple combination method. On a single database scan, a sequence id is taken and all possible combination of sequences is generated. Once the sequences are generated, the sequences are stored in a tree. The nodes of the tree also contain frequency of that sequence. If a same sequence exists in single sequence id, its count will not be increased. The count will only be increased if that sequences will be present in some other sequence id. A sequence can be searched with the help of the following steps:

- Take the sequence which you want to search.
- Take first item of the sequence and search it in the tree.
- If the first item is matched, then search in downward child nodes for the sequence.

After we are done with all sequence ids, we will look for nodes that will satisfy the minimum support. The nodes which will not satisfy the minimum support will be pruned and we will be with nodes which are frequent.

4 Illustrative Example

First, we take sequence id S1 and make all possible sequences. For example, suppose we take first sequence id S1 of sequence database given in Fig. 1.

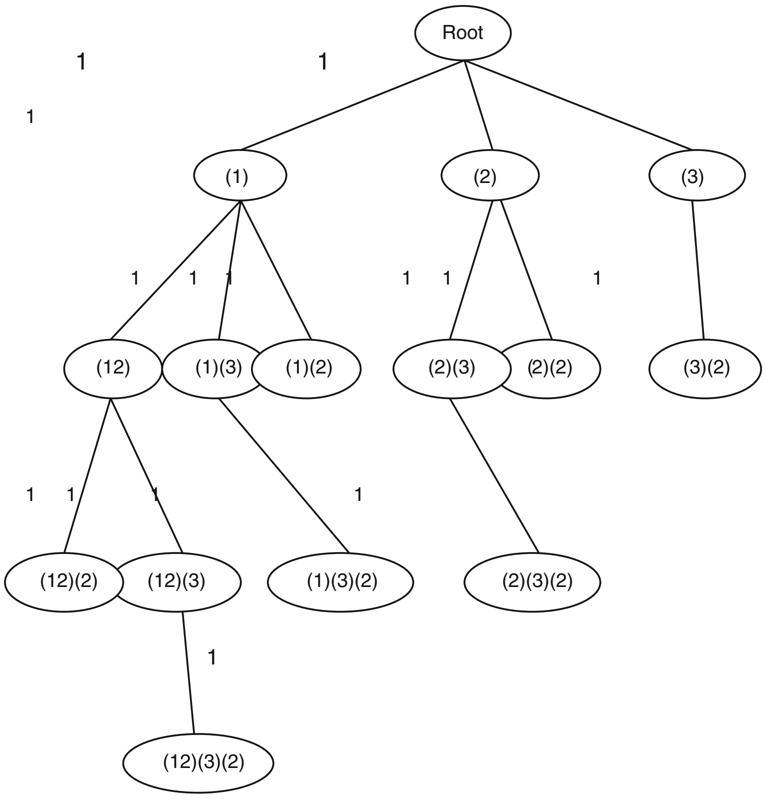


Fig. 1 Tree formed by sequences of sequence id S1

S1 (1 2) (3) (2)

The sequences generated from this sequence are as follows:

(1), (2), (3), (1 2), (1)(3), (1)(2), (2)(3), (2)(2), (3)(2), (1)(3)(2), (2)(3)(2), (1 2)(3), (1 2)(2), (1 2)(3)(2).

The sequences that are generated are placed in the form of tree given in Fig. 1.

Root will be at level 0, 1-sequence sequences will be stored at level-1, and so on. Figure 1 shows the tree formed by the sequences of S1. The count 1 indicates the frequency of all the sequences. Now we will take sequence S2, generate all sequences, and place them in the same tree shown in Fig. 1. If the sequence is already present in the tree, then count of the node will be increased; otherwise the sequence will be placed in the tree.

After S1 is stored in tree, now we take S2. All the sequences formed by S2 are S2 (3)(2 3)

(3), (2), (3)(2), (3)(3), (2 3), (3)(2 3).

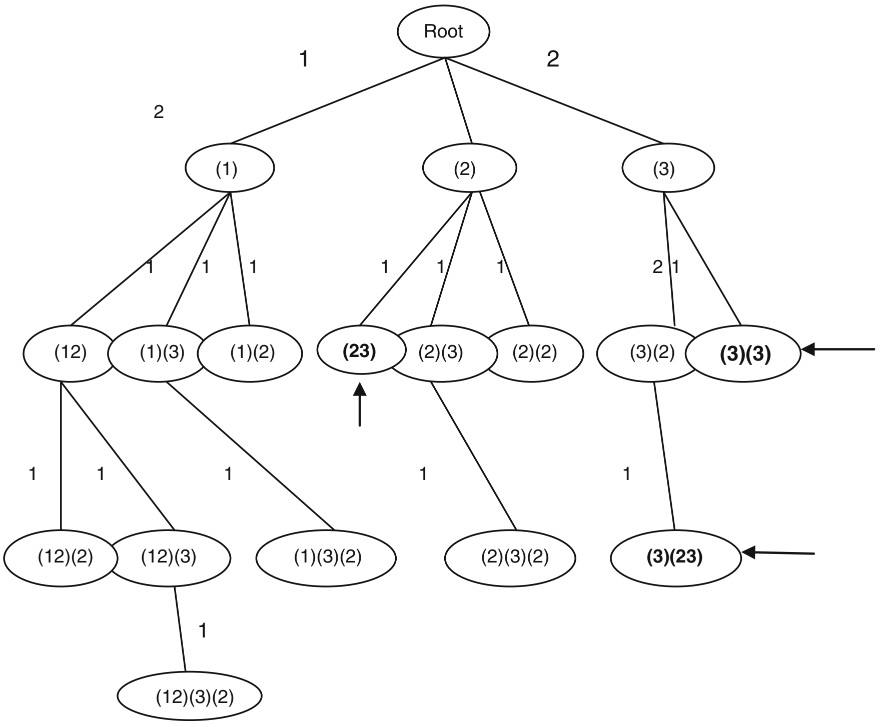


Fig. 2 Tree formed after storing sequences of S2

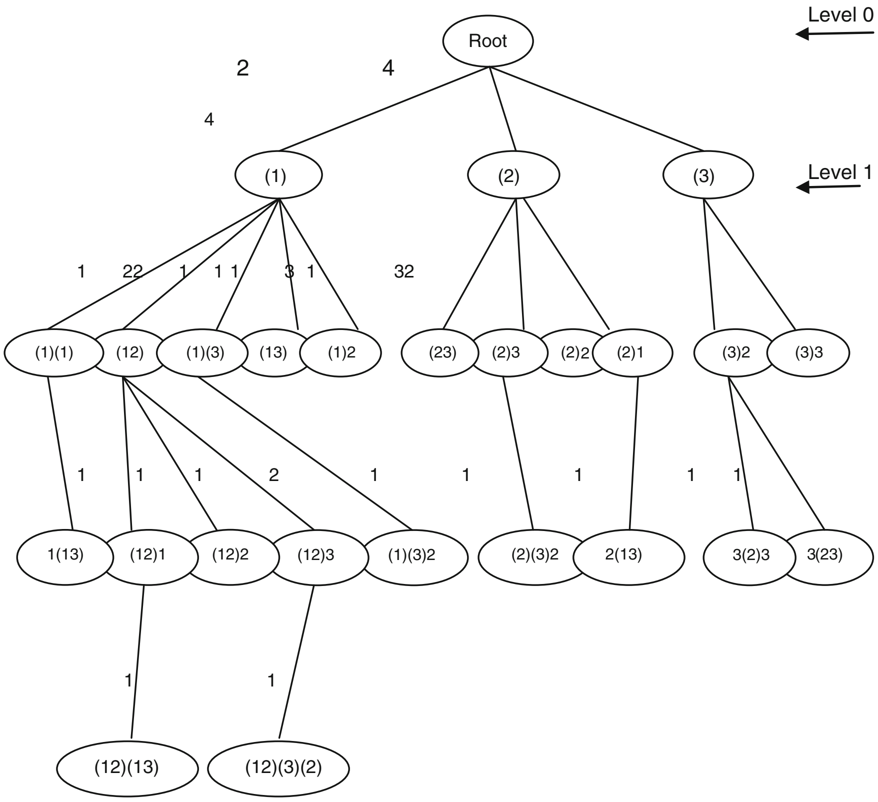
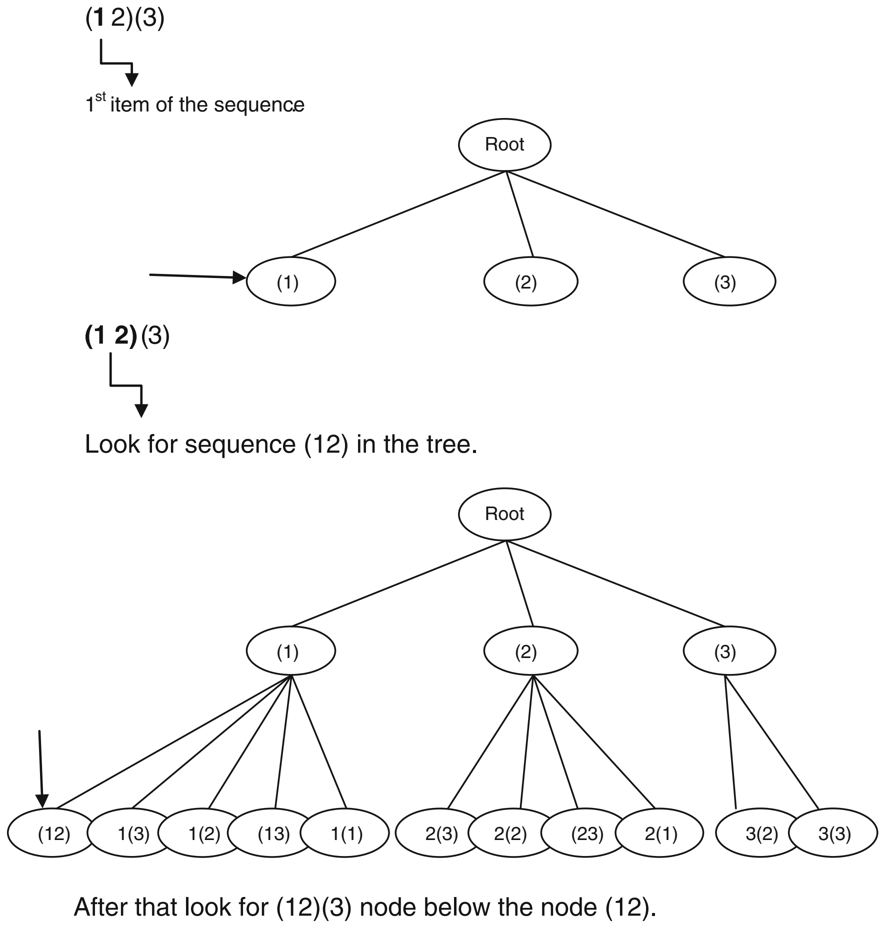


Fig. 3 Final tree formed by sequence database given in Table 1

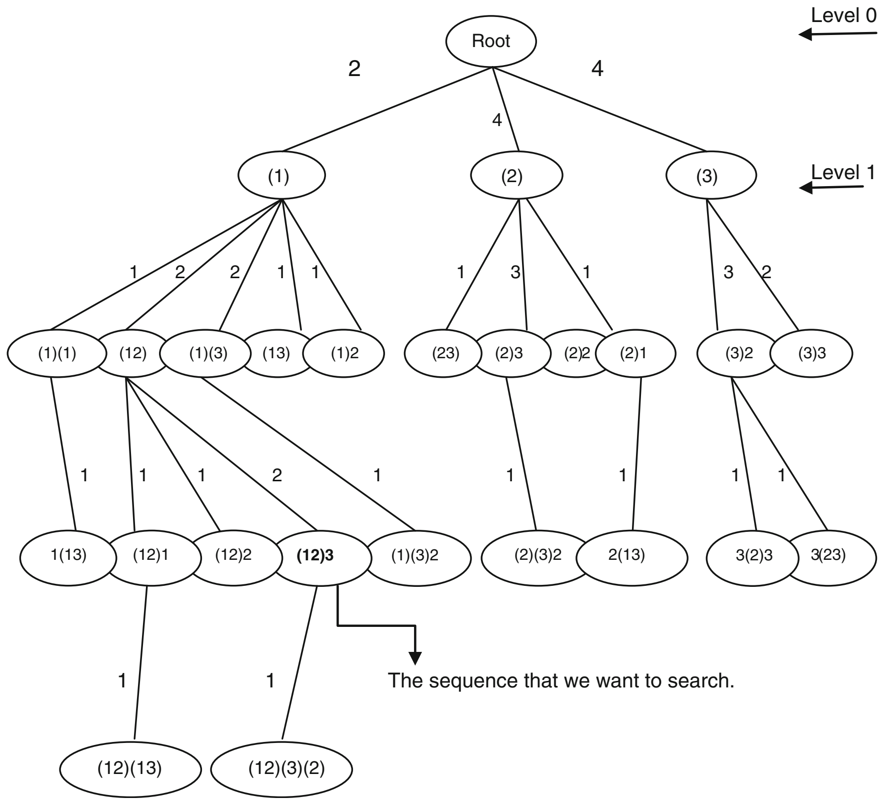
Sequences such as (2), (3), and (3)(2) already exist in tree in Fig. 1, so their counts are increased to 2, whereas sequences (3)(3), (2 3), and (3)(2 3) are stored as shown in Fig. 2. Same procedure is applied for all sequences (Fig. 3).

Searching any sequence in the tree is also very easy and simple, since all the nodes save the path that makes easy for us to know all the nodes above that particular node. Now suppose we have to search sequence (1 2)(3). The sequence (1 2)(3) is a 3-sequence, so it will be at level-3. According to the searching technique, since the first item of them sequence (1 2)(3) is 1, we will look for the nodes under node (1) in the tree. Afterward, we will look for the second item of the sequence, i.e., (1 2).

So we will look for (1 2) within same bracket, and finally we will look for nodes under node (1 2).



After that look for (12)(3) node below the node (12).



By this way storing complete previous path on each and every node makes the searching of sequence very easy; since we do not have to backtrack to search for a particular, this provides us an idea of where a particular sequence might be stored.

So the frequent sequential patterns of sequence database are shown in Tables 1 and 2.

Table 1 Sequence database

Sequence Id	Sequences
S1	(1 2)(3)(2)
S2	(3)(2 3)
S3	(1 2)(1 3)
S4	(3)(2)(3)

Table 2 All frequent sequential patterns

Sequence	Count
(1)	2
(2)	4
(3)	4
(1 2)	2
(1)(3)	2
(2)(3)	3
(3)(3)	2
(3)(2)	3
(1 2)(3)	2

5 Conclusion

The algorithm proposed in this paper generates the complete set of frequent sequential patterns without generating any candidates who reduce both the times. It also reduces the effort of repeated database scanning, thereby improving the performance of algorithm. The algorithm stores both frequent as well as non-frequent items. Although storing of non-frequent items require little more memory, but in future better performance will be achieved for incremental mining because of stored non-frequent items. We can ignore memory usage issue for performance enhancement benefits as nowadays memory is not so expensive, and will also be shown later.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of 20th International Conference on Very Large Databases (1994)
2. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th International conference on Extending Database Technology (EDBT'96), Avignon, France, September, pp. 3–17 (1996)
3. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1–2), 31–60 (2001)
4. Masseglia, F., Poncelet, P., Teisseire, M.: Using data mining techniques on web access logs to dynamically improve hypertext structure. ACM Sig. Web Lett. **8**(3), 13–19 (1999)
5. Han, J., et al.: FreeSpan: frequent pattern-projected sequential pattern mining. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge discovery and data mining, ACM (2000)
6. Pei, J., et al.: PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), IEEE Computer Society (2001)
7. Ayres, J., et al.: Sequential pattern mining using a Bitmap Representation. In: Proceedings of Conference on Knowledge Discovery and Data Mining, pp. 429–435 (2002)
8. Kaneiwaa, K., Kudob, Y.: A sequential pattern mining algorithm using rough set theory. Int. J. Approximate Reasoning **52**(6), 881–893 (2011)

9. Nakamura, S., Nozaki, K., Norimoto, Y., Miyadera, Y.: Sequential pattern mining method for analysis of programming learning history based on learning process. In: The International Conference on Educational Technologies and Computers (ICETC), IEEE, ISBN: 978-1-4799-647-1, September 2014, pp. 55–60 (2014)
10. Wright, A.P., Wright, A.T., McCoy, A.B., Sittig, D.F.: The use of sequential pattern mining to predict next prescribed medications. J. Biomed. Inf. **53**, 73–80 (2015)