Fast Accumulation Lattice Algorithm for Mining Sequential Patterns

NANCY P. LIN, WEI-HUA HAO, HUNG-JEN CHEN
Department of Computer Science and Information Engineering
Tamkang University
151 Ying-chuan Road, Tamsui, Taipei,
TAIWAN

Abstract: - Sequential Patterns has many diverse applications in many fields recently. And it has become one of the most important issues of Data Mining. The major problem in previous studies of mining sequential patterns is too many candidates sequences has been generated during the mining process, costing computing power and increasing runtime. In this paper we propose a new algorithm, Fast Accumulation Lattice (FAL) to alleviate this problem. FAL scan sequential database only once to construct the lattice structure which is a quasi-compressed data representation of original sequential database. The advantages of FAL are: reduce scan times, reduce searching space and minimize requirement of memory for searching frequent sequences, and maximal frequent sequences as well.

Keywords: - sequential patterns, sequence mining, data mining, maximal frequent sequence

1 Introduction

Frequent Sequences Mining is an important task of data mining, in the point of view of applications, including Learning Patterns, Web access patterns, Customer behavior analysis and others time related data process. The problem can be state as *Sequential pattern mining is to discover frequent subsequences as patterns in a sequence database* [11].

There are many previous studies of mining sequential patterns efficiently [2][3][5][8]. Most, almost all, of the previous studies of mining sequential patterns, time related sequence, are adopting apriori-like principle which denote that any super-sequence of an infrequent sequence is also infrequent. The apriori-like principle is based on generation-and-prune method; the first scan is finding all of the frequent, also know as *large* in association rule mining, 1-sequence and which is assembled to generate 2-sequence candidates. Those candidates not satisfying the minimum support threshold will be pruned in the mining process. Repeat the process until no more candidates were generated.

The apriori-like sequential pattern mining methods has suffered from several major drawbacks: (1) generate a huge set of candidates from a sequence database, (2) poor time efficiency due to multiple scans of sequence database, (3) exponentially generating combinational candidate sequences in the process of long sequential patterns mining; low threshold as well.

In this paper, we propose a novel algorithm, FAL, differ from previous studies. Main goal is to reduce searching space and runtime via a lattice structure algorithm.

The remainder of this paper is organized as follows: In section 2, preliminary concepts are introduced. In the section 3, the Maximal Sequence is defined. In Section 4, our algorithm, Fast Accumulation Lattice, is introduced. In Section 5, the FAL algorithm is illustrated with an example. The conclusion is in the Section 6.

2 Preliminary

Let $I=\{i_1,i_2,...,i_n\}$ be a set of all items. A subset of I is called an itemset. A sequence is defined as $s=\langle e_1,e_2,...,e_m\rangle$, $e_1\subset I, e_2\subset I,...,e_m\subset I$. The length of a sequence is the total number of items in the sequence. If a sequence $a=\langle a_1,a_2,...,a_m\rangle$ *contains* sequence $b=\langle b_1,b_2,...,b_n\rangle$ then we denote this relationship as $a\prec b$, if and only if $\exists i_1,i_2,...,i_m$, and that $1\le i_1<i_2<...<i_m\le n$ and $a_1\subseteq b_{i_1}$, $a_2\subseteq b_{i_2}$,…, $a_m\subseteq b_{i_m}$. We denote that sequence *a* is a *subsequence* of *b*, *b* is *supersequence* of *a*. A sequence database is a set of sequences. Suppose *D* is a sequential database. |D| is representing the number of sequences in the sequence database *D*. The *support* of a sequence *s* is the number of sequences in D which contains sequence *s*. A minimum support, *minsup*, is the threshold of frequent sequence. All sequences with support no less than *minsup* are called frequent sequence, *FS*. The Maximal Frequent Sequences, *MS*, is one of the outputs of FAL. The set of maximal frequent sequence is defined as $MFS=\{s\mid s\in FS \text{ and } \neg\exists s'\in FS \text{ such that } s\prec s'\}$.

The main issue of mining maximal frequent sequence is to find *MFS* with support no less than minimum support threshold.

In this paper we introduced upward closure principle and downward closure principle. The upward closure principle is also known as the ariopri principle; all supersequences of an infrequent sequence are also infrequent. The downward closure indicates that all subsequences of a frequent sequence are also frequent. In our FAL algorithm will apply these two principles as an essential part of algorithm.

3 Maximal Sequences

In previous studies of sequential patterns mining algorithms focus on mining the full set of frequent subsequences that support no less than a given minimum support in a sequence database. On the other hand, a frequent long sequence contains a combinatorial number of frequent subsequences, cost expensively in both time and space, is the major inevitable drawback. A maximal sequence approach was introduced to meet these problems.

Let *S*, $S\subset T$ and $S=\{s_1,s_1,\cdots,s_n\}$, be a set of all frequent sequences. If we delete all subsequences of each sequence of *S*; the remaining sequences are called Maximal Sequences. In a sense, MS is the compressed representation of FS. This result is due to the downward closure principle. In many cases, the idea of maximal sequences is recommended to represent the whole frequent subsequences for the sake of condense information. For instance, a maximal frequent n-sequence has m subsequences, $m=2^n-1$. The condense rate is m, which means that space of maximal frequent sequence is m times smaller than the whole set of its subsequences. The longer the maximal frequent sequence the better condense rate will be. With maximal sequences, the lattice can be more easily fit into memory.

To our percept, a structure of maximal frequent sequences is a lossless method to contain original information.

4 Fast Accumulation Lattice Algorithm

In this section, we described the concept of Fast Accumulation Lattice. FAL has 3 sequential phases: Growing Phase, Pruning Phase and Maximal Phase. In Growing Phase, all sequences are read from database into a lattice data structure. Each node in the lattice accumulates the count of sequences and passes the count to all its subsequences nodes. Those nodes with count no less than *minsup*, and all its subsequences nodes, will be marked as frequent sequence node. So, to speed up the FAL algorithm it doesn't have to pass the accumulating count to these marked nodes. Second, in Pruning Phase, prune off those infrequent sequence nodes in the lattice. Finally, Maximal Phase, delete each node's subsequences to find out maximal frequent sequences.

In this paper we applied the idea of maximal sequence in FAL to improve the efficiency problem of apriori-like sequence mining.

Algorithm FAL(*D, minsup*)

```
//Input: D, minsup
//Output: MFSMaximal Sequences
// Growing Phase
Initiate Lattice   // create root node
ConstructLattice(D,minsup){
    Repeat until the end of D{
        read sequence SP from D
        if SP ∈ Lattice {
```

```
                search node that node.sequence=SP;
                if node.frequent=FALSE{
                    node.count++;
                    if node.count≥minsup
                        For this node and all of it's
                        subsequence
                        node.frequent=TRUE;
                }
            }
            else {
                new node;
                node.sequence=SP;
                node.count=1;
                //construct all subsequences nodes of
                this new node;
                ConstructLattice(subsequences of
                node, minsup);
            }
        }
}
//Pruning Phase
For (k=1:k≤ |MaxFrequentSequence|-1 :k++)
    For all nodes of length k
        If node.frequent=FALSE{
            Delete node;
            Delete all supersequences nodes
        }
// Maximal sequence phase
For each node{
    Delete all subsequences nodes
    Maximal sequences = all nodes remain in the
    Lattice
    }
```

5 example

For example, table 1 is a learning sequence database table. SID represents Sequence Identifier. In this database has included only 5 items of A, B, C, D and E. In this example, the length of the longest sequence is 4. We set *minsup* =2 in this example.

Table 1 sample sequence database

SID	Sequence
1	ACD
2	ABCE
3	BCE
4	BE

GROWING PHASE: read in the first sequence 〈ACD〉 from database. Link the sequence node to root node and all its subsequence are linked to this node and so on so forth. This lattice is grow into a 8 nodes lattice including one root node, one 3-item sequence{〈ACD〉}, 3 2-item sequences {〈AC〉, 〈AD〉, 〈CD〉}, 3 1-item sequence {〈A〉, 〈C〉, 〈D〉}. The accumulator of each node has increased by 1 from 0. The lattice is shown as Fig.1.

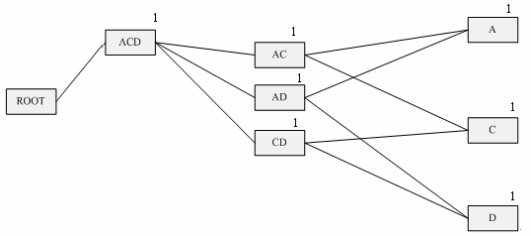

Fig. 1 Lattice with 〈ACD〉 and subsequences

After read in the second sequence 〈*ABCE*〉 from sequence database, the first step is to check whether 〈*ABCE*〉 is already exist in the lattice, not in the lattice or it is a subsequence of a node in the lattice. Second, find out the common subsequence of each node of the lattice, for now only 〈*ACD*〉, and 〈*ABCE*〉. In this example the common sequence is 〈*AC*〉. The accumulators of node 〈*AC*〉 and all its subsequence are counted as 2 which has reached the minimum support. So far, there are 3 frequent sequence nodes in the lattice. The lattice structure is shown in Fig.2. Since 〈*AC*〉, 〈*A*〉 and 〈*C*〉 are frequent sequence node these 3 nodes are shown as white node. Gray node represents infrequent sequences. The number on the upper right corner represents accumulated count number.

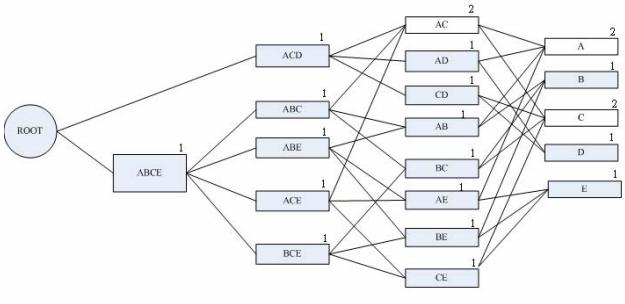

Fig. 2 Lattice of 〈ACD〉 and 〈ABCE〉

The Lattice after reading 〈BCE〉and〈BE〉 is shown in fig.3. White nodes, 〈AC〉〈A〉〈C〉 〈BCE〉〈BC〉〈BE〉〈CE〉〈B〉and 〈E〉 ,are denoted as frequent sequences since their accumulator has count number no less than *minsup*.

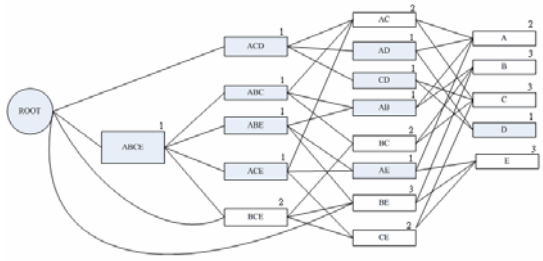

Fig. 3 Complete Lattice

Pruning Phase: Scan Lattice from short sequence nodes, delete infrequent sequence nodes. A pruned Lattice is shown in Fig.4。

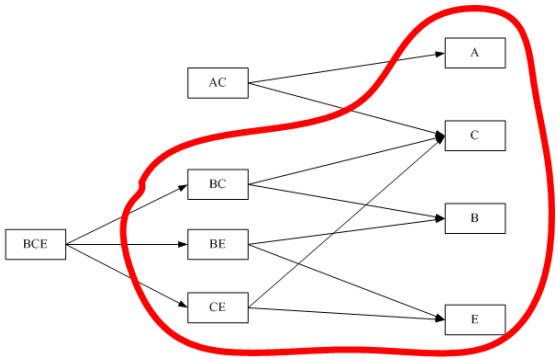

Fig. 4 Lattice of frequent sequence

Maximal Sequences: Scan from long sequence nodes, delete each node's subsequence, surrounded with a big red circle in Fig. 4. The remaining sequences 〈BCE〉and 〈AC〉 are called maximal sequences. Lattice of MFS is shown as Fig.5.

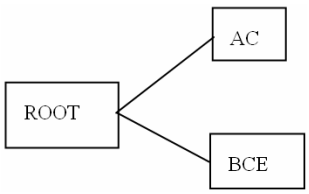

Fig. 5 Lattice of Maximal Sequences

The advantages are: Small search space and faster search speed via the pre-knowledge of minimum support.

6 Conclusion

In this paper we had introduced a novel lattice structure to represent the original sequence database in a sense of compress method. We also investigated issues for mining maximal frequent sequential patterns in sequence database and highlight the essential problem of possible inefficiency and redundancy of mining frequent sequential patterns. To the best of our knowledge, this is the first study to solve maximal frequent sequences problem with lattice structure. In theory, FAL is outperform the previous apriori-like algorithm for mining sequential patterns by lot more less space occupation, searching space and runtime.

The new algorithm consists of three major phases; (1) growing phase: scan the sequence database to build a lattice structure with a given minimum support threshold and take the advantage of this pre-knowledge to accelerate the building speed and achieve a more compact structure. (2) pruning phase : prune all infrequent sequences via apriori principle, start from short sequence, delete all infrequent sequence and all it's supersequences. (3) maximal sequence phase: for each sequence in FS delete all it's subsequences, the remaining is called MS.

Reference:

[1] Jiawei Han and Micheline Kamber, "Data Mining, Concepts and Techniques", 2nd edition, Morgan Kaufmann Published, 2006.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In Proc. 1995 Int. Conf. Data Engineering (ICDE'95), pages 3–14, Taipei, Taiwan, Mar. 1995.

[3] J. Han, J. Pri, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining, Proc. 2000 ACM SIGKDD Int'l Conf. Knowledge Discovery in Database (KDD '00), pp. 355-359, Aug. 2000.

[4] J. Han, J. Pei and Y. Yin, "Mining Frequent Patterns without Candidate Generation", Proc. 2000 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '00), pp.1-12, May 2000.

[5] Wang, J.; Han, J.a, "BIDE: efficient mining of frequent closed sequences", Data Engineering, 2004. Proceedings. 20th International

Conference on 30 March-2 April 2004 Page(s):79 – 90.

[6] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discoving frequent closed itemsets for association rules. In ICDT' 99, Jerusalem, Israel, Jan. 1999.

[7] J. Wang, J. Han, and J. Pei, CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In KDD ' 03,Washington, DC, Aug. 2003.

[8] X. Yan, J. Han, and R. Afshar," CloSpan: Mining Closed Sequential Patterns in Large Databases". In SDM' 03, San Francisco, CA, May 2003.

[9] M. Zaki, and C. Hsiao, CHARM: An efficient algorithm for closed itemset mining. In SDM' 02, Arlington, VA, April 2002.

[10] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Janyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Mei-Chun Hsu, Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach, IEEE Transactions on Knowledge and Data Engineering, vol. 16, No. 11, November 2004.

[11] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning, 40:31–60, 2001.

[12] Maged El-Sayed, Carolina Ruiz, Elke A. Rundensteiner, Web mining and clustering: FS-Miner: efficient and incremental mining of frequent sequence patterns in web logs Proceedings of the 6th annual ACM international workshop on Web information and data management, November 2004.

[13] R. Agrawal and R. Srikant,Fast Algorithms for Mining Association Rules, Proc. 1994 Int'l Conf. Very Large Data Bases(VLDB '94), pp.487-499. 1994.

[14] R. Agrawal and R. Srikant, Mining Sequential Patterns, Proc. 1995 Int'l Conf. Data Eng. (ICDE '95), pp.3-14, Mar. 1995.