Fast Mining of Closed Sequential Patterns

NANCY P. LIN, WEI-HUA HAO, HUNG-JEN CHEN, HAO-EN CHUEH, CHUNG-I CHANG
Department of Computer Science and Information Engineering
Tamkang University,
151 Ying-Chuan Road, Tamsui, Taipei,
TAIWAN, REPUBLIC OF CHINA
nancylin@mail.tku.edu.tw, 889190111@s89.tku.edu.tw, chenhj@mail.sju.edu.tw,
890190134@s90.tku.edu.tw, taftdc@mail.tku.edu.tw

Abstract:- This paper propose a novel algorithm for mining closed frequent sequences, a scalable, condensed and lossless structure of complete frequent sequences that can be mined from a sequence database. This algorithm, FMCSP, has applied several optimization methods, such as equivalence class, to alleviate the needs of searching space and run time. In particular, since one of the main issues in this type of algorithms is the redundant generation of the closed sequences, hence, we propose an effective and memory saving methods, different from previous works, does not require the complete set of closed sequences to be residing in the memory.

Key-Words: - data mining, sequential patterns mining, closed sequential patterns

1 Introduction

Mining sequential patterns is a task of finding the full set of frequent sequences that satisfy a given minimum support in a sequence database. Sequential pattern mining was introduced in [2], has gradually become an essential data mining task, with broad applications, including market and customer analysis, web log analysis, pattern discovery in protein sequences, and mining XML query access patterns for caching.

In this paper, we propose a solution to mine closed sequences, rather than mining the full set of frequent sequences. Closed sequences are the unique maximal sequences of the equivalence classes.

The problem can be described as follows: Assume that $\boldsymbol{I}=\{ i_1, i_2,\ldots, i_{|\boldsymbol{I}|}\}$ is a finite items set and $\boldsymbol{D}$ is a data set containing n transactions, each transaction $s \in D$ is a sequence of distinct items $s= \langle i_1,\ldots,i_{|s|} \rangle$, in which $i_j \in \boldsymbol{I}$. Let S be a *k-items* sequence, where $S = \langle i_1,\ldots,i_k \rangle$ is a sequence of k distinct items $i_j \in \boldsymbol{I}$. Given a *k-items* sequence s, let its support be supp(s) which is defined as the number of transactions in $\boldsymbol{D}$ that include s. To mine all the frequent sequences from $\boldsymbol{D}$ requires finding all the sequences that support no less than the minimum support and this has to search through the huge search space which is given by the power set of $\boldsymbol{I}$. Han and Kamber[17] defined *:A sequence database consists of sequences of ordered elements or events.*

Sequential pattern mining has been studied extensively in recent years. Most previous studies are required to specify a minimum support threshold to perform the mining procedure. Usually, in practice, it is difficult for knowledge workers to provide an appropriate threshold previously. The reason why we mine closed sequential patterns is that they are compact representations of frequent sequential patterns. Further, we proposed an efficient algorithm, called FMCSP, which makes use of the minimum support constraint and the properties of closed sequential patterns to perform dynamic support searching and database pruning.

Sequential Patters Mining was first introduced by Agrawal and Srikant in [1]:

Given a set of sequences, where each sequence consists of a list of itemsets, and given a user-specified minimum support threshold (min support), sequential pattern mining is to find all frequent subsequences whose frequency is no less than min support. This mining algorithm has a consequence of the following two problems.

First, sequential pattern mining often generates huge number of candidate patterns in an exponential curve, which is inevitable when the database consists of long frequent sequential patterns. For example, assume the database contains a frequent sequence $\langle i_1,\ldots,i_k \rangle$, k=20, it will generate 2^{20} -1 frequent subsequences which are essentially redundant patterns.

Second, setting minimum support is also a difficult job for knowledge works: The smaller value of minimum support may lead to generate larger amount of candidate sequential patterns, whereas a too big one may cause no answer found.

In practice, to get an appropriate minimum support, one needs to have domain knowledge about the data, and be able to estimate the scale of how many patterns will be generated within a mining process with a particular threshold of minimum support.

A sequence *s* is called *closed* if there exists no super sequence of *s* with the same support in the database. Several studies to the first problem was proposed recently by Yan, et al. [9], Han, et al. [6]. Their algorithm can mine closed sequential patterns. Mining sequential patterns with closed patterns may largely reduce the number of patterns generated in the process and without losing any information because it can be used to derive the complete set of sequential patterns.

As to the second problem, as proposed in [5], a solution is to change the task of mining frequent patterns to mining top-k frequent closed patterns of minimum length *min-l*, where *k* is the number of closed patterns to be mined, top-*k* refers to the *k* most frequent patterns, and *min-l* is the minimum length of the closed patterns. This setting is also desirable in the context of sequential pattern mining. Unfortunately, most of the techniques developed in [5] cannot be directly applied in sequence mining. This is because subsequence testing requires order matching which is more difficult than subset testing. Moreover, the search space of sequences is much larger than that of itemsets. Nonetheless, some ideas described in [5] are still influential in our algorithm. Mining closed sequences [6] has indicated that previous studies have presented that *a frequent pattern mining algorithm should not mine all frequent patterns but only the closed ones because the latter leads to not only more compact yet complete result set but also better efficiency.* However, most of the previously developed algorithms of closed pattern mining work under the candidate generate-and-prune methodology which is inherently costly in searching space, space usage and runtime when the support threshold is low or the patterns become long.

In recent years many studies of sequence mining have concluded a valuable point of view that for both itemsets and sequences, one should not mine *all* frequent patterns for saving both space and runtime. But the *closed patterns*, itemsets and sequences, are not only more compact and complete set but also better efficiency [7,8,9,10]. However, unlike mining frequent itemsets have been studied heavily, there are not so many methods proposed for mining closed sequential patterns. Like most of the frequent closed itemset mining algorithms, CloSpan [9] follows a *candidate maintenance-and-test* paradigm, i.e., *it needs to maintain the set of already mined closed sequence candidates which can be used to prune search space and check if a newly found frequent sequence is promising to be closed.* Unfortunately, under such a paradigm a closed pattern mining algorithm has rather poor scalability because a large number of frequent closed patterns, or candidates, will need large memory and lead to large search space for the closure checking of new patterns, when the support threshold is low or the patterns are long. Can we find a way to mine frequent closed

sequences without candidate maintenance? This seems to be a very difficult task.

2 Preliminary

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of distinct items. A sequence S is an ordered list of events, denoted as $\langle e_1, e_2, \ldots, e_m \rangle$, where e_i is an item, i.e., $e_i \in I$ for $1 \le i \le m$. For brevity, a sequence is also written as $e_1e_2 \ldots e_m$. From the definition we know that an item can occur multiple times in different events of a sequence. The number of events (i.e., instances of items) in a sequence is called the length of the sequence and a sequence with a length l is also called an l-sequence. For example, *ABCD* is a 4-sequence.

A sequence $S_a = a_1 a_2 ... a_n$ is contained in another sequence $S_b = b_1 b_2 ... b_n$, if there exist integers $1 \le i_1 < i_2 < ... < i_n \le m$ such that $a_1 = b_{i_1}, a_2 = b_{i_2}, ..., a_n = b_{i_n}$. If sequence S_a is contained in sequence S_b, S_a is called a subsequence of S_b and S_b a supersequence of S_a, denote as $S_a \prec S_b$. An input sequence database D is a set of tuples (*sid*, S), where *sid* is a sequence identifier, and S an input sequence. The number of tuples in D is called the base size of D, denoted as $|D|$. A tuple (*sid*, S) is said to contain a sequence S_α, if S is a supersequence of S_α, i.e., $S_\alpha \prec S$. The *absolute support* of a sequence S_α in a sequence database D is the number of tuples in D that contain S_α, denoted as *supp* (S_α), and the *relative support* is defined as the percentage of tuples in D that contain S_α

$$relative\_support = \frac{supp(S_\alpha)}{|D|}$$

Let T and S, $T \subseteq D$ and $s \subseteq I$, be subsets of all the transactions and sequences appearing in D. The following two functions can describe the concept of closed sequences:

$f(T) = \{s \in S \mid \forall t \in T, s \in t\}$ Find all sequences in T

$g(S) = \{t \in D \mid \forall s \in S, s \in t\}$ Find all transactions in S

Definition 1. An sequence S is closed *iff*

$$c(S) = f(g(S)) = f \circ g(S) = S$$

The composite function is named as the closure operator. The closure operator defines the equivalence class of sequences.

Definition 2. For itemsets belongs to the same equivalence class *iff* they are supported by the same set of transactions.

3 The FMCSP Algorithm

Initially, FMCSP has no need to input the threshold of *min_sup*. First, create Lattice with FAL algorithm [16]. In the meaning time, itemset is generated during the FAL process. In the end, equivalence class table, ECT, is generated and sorted into a projected ECT. For now, the sequential database D is lossless and completely represented by the lattice, ECT and Projected-ECT.

Algorithm FMCSP (*D*)

```
//Input: D
//Output: Frequent Sequencesf
int min_sup;
CreateLattice(D);
ItemSet=GenerateItemset(D);
ECT=GenerateClassTable(ItemSet);
PECT=Projected(ECT);
cin>>min_sup
FrequentSequences= Tuples in PECT with Frequency noless than min_sup
```

Fig. 1

It is convenient to demonstrate the idea of closure operator with example. In table 1, as a simple transaction database, there are 6 sequences with diverse length from 1 to 4. Items includes *A*, *B*, *C*, *D*, *E*, *F* and *G*, totally 7 items.

Table 1 example sequence

SID	sequence			
1	*B*	*D*		
2	*A*	*B*	*C*	*D*
3	*A*	*C*	*D*	
4	*C*			
5	*A*	*B*	*E*	
6	*F*	*G*		

Function of $f(1)$, $f(2)$, $f(3)$, $f(4)$, $f(1,2)$, $f(1,3)$ and $f(1,4)$ are shown as below:

.

$$f(1) = BD$$
$$f(2) = ABCD$$
$$f(3) = ACD$$
$$f(4) = C$$
$$f(5) = ABE$$
$$f(6) = FG$$
$$f(1,2) = BD$$
$$f(1,3) = D$$
$$f(1,4) = C$$
$$f(2,3) = ACD$$

Fig.2

Functions of $g(A)$, $g(B)$, $g(C)$,…, $g(G)$ and $g(AB)$, $g(AC)$,…, $g(CD)$ and $g(ABC)$, $g(ABD)$,…, $g(BCD)$ and $g(ABCD)$ are shown as below:

$$g(A) = \{2,3,5\}$$
$$g(B) = \{1,2,5\}$$
$$g(C) = \{2,3,4\}$$
$$g(D) = \{1,2,3\}$$
$$g(E) = \{5\}$$
$$g(F) = \{6\}$$
$$g(G) = \{6\}$$
$$g(AB) = \{2,5\}$$
$$g(AC) = \{2,3\}$$
$$g(AD) = \{2,3\}$$
$$g(AE) = \{5\}$$
$$g(BC) = \{2\}$$
$$g(BD) = \{1,2\}$$
$$g(BE) = \{5\}$$
$$g(CD) = \{2,3\}$$
$$g(ABC) = \{2\}$$
$$g(ABD) = \{2\}$$
$$g(ABE) = \{5\}$$
$$g(ACD) = \{2,3\}$$
$$g(BCD) = \{2\}$$
$$g(ABCD) = \{2\}$$
$$g(FG) = \{6\}$$

Fig.3

In this example, there are 10 equivalence classes {2}, {5}, {6}, {1,2}, {2,3}, {2,5}, {1,2,3}, {1.2.5}, {2,3,4} and {2,3,5} supported by different transactions. In {1,2} equivalence class there is only one sequence 〈 *BD* 〉, it means in the set of sequences only sequence 〈 *BD* 〉 is supported by transaction 2 (TID=2) and so on so forth.

In this example, the whole database is converted into a Equivalence Class Table (ECT), shown as table 2, and sorted by frequency in ascendant order. The first column is the series number of equivalent class. The second column is the title of the equivalent class. The third column is the element sequences of each equivalent class. The last column is the frequency, counts, of each equivalent class.

Table 2 Equivalence Class Table (ECT)

No.	Equivalent class	SEQUENCE					Frequency
1	{2}	*BC*	*ABC*	*ABD*	*BCD*	*ABCD*	1
2	{5}	*ABE*	*AE*	*BE*	*E*		1
3	{6}	*FG*	*F*	*G*			1
4	{1,2}	*BD*					2
5	{2,3}	*AC*	*AD*	*CD*	*ACD*		2
6	{2,5}	*AB*					2
7	{1,2,3}	*D*					3
8	{1,2,5}	*B*					3
9	{2,3,4}	*C*					3
10	{2,3,5}	*A*					3

The size of search space is the number of equivalent class, in this example, is 10.

It is convenient to project the ECT from ECT into frequency indexing table, as shown in Table 3, to speed up searching runtime of a certain support equivalence class.

Table 3 Projected ECT (PECT)

Frequency	Serial No.			
1	1	2	3	
2	4	5	6	
3	7	8	9	10

The search space, with projected view, has been limited to only 3 tuples.

Let minimum support, *min_sup*, equals to 3, the frequent sequences are <D>, <B>,<C> and <A>. Let *min_sup*=2, the frequent sequences are <BD>, <AC>, <AD>, <CD>, <ACD>, <AB> and <D>, <B>,<C>, <A>.

With FMCSP, the minimum support threshold is no longer limited to a prefixed variable which means FMCSP a more flexible algorithm than our previous works [16]. For example, sequence *AC* is tested by closure function as below:

$$f(g(AC)) = f(2,3) = ACD$$

Obviously, sequence *AC* is not equal to sequence *ACD*, so sequence *AC* is not a closed sequence. In other example of sequence *BD*, tested with closed function as below

$$f(g(BD)) = f(1,2) = BD$$

Shows the result of closed function is the same as the input sequence, *BD*, so sequence *BD* is a closed sequence. It is also clear that a sequence is closed if and only if no *supersequences* of S with the same support exist in the lattice. Each equivalence class contains elements sharing the same supporting transactions and closed sequences are their maximal elements.

We demonstrate the construction of Lattice with FAL algorithm via example. The first sequence read from database linked to root node is <BD>. Shown as fig. 4.

Fig.4

Next sequence read from sequence database is <ABCD>. Since there is no sequence in the lattice that contains sequence <ABCD>, so it is linked to root the node, as a sibling node of sequence <BD>, shown as fig. 5. Sequence node <BD> and <ABCD> are highlight with bold frame to indicate that these two sequence are read from original sequence database.

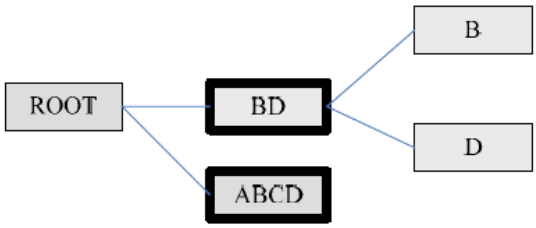

Fig.5

But, the new sequence <ABCD> contains the sequence <BD> that already exists in the lattice. So sequence node <BD> has to reconnect its upper link to sequence node <ABCD>. After next two consecutive sequences, <ACD> and

<C>, are read from sequence database. The lattice is attached with sequences <ACD> and <C>, shown as fig. 6.

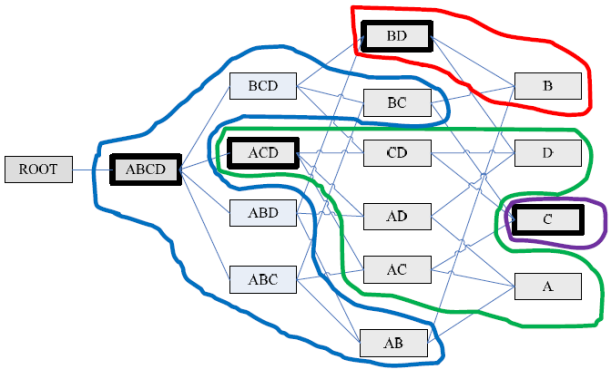

Fig.6

Finally, the last two sequences, <ABE> and <FG> were linked into the lattice. In Fig. 7, 10 closed sequences were framed with broad rectangle. Each closure represented an equivalence class.

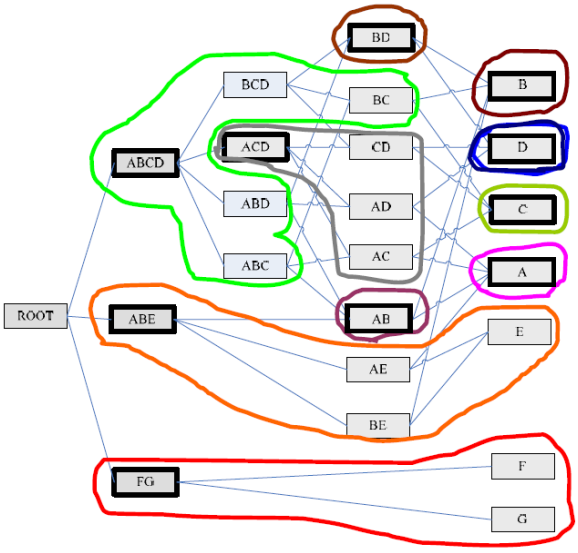

Fig. 7 Lattice with 10 equivalence classes

4 Conclusion

Unfortunately, Apriori-like algorithms may fail to extract all the frequent sequences from dense data sets, which contain strongly correlated sequences and long frequent sequential patterns. Such data sets are, in fact, very hard to mine since the Apriori closed-downward principle does not guarantee an effective pruning of candidates, while the number of frequent sequences grows up very quickly as the minimum support threshold is decreased.

Many studies have incept the concept to elaborate all frequent pattern mining to more compact results and significantly better efficiency of memory usage. Our study shows that this is usually true when the number of frequent patterns is extremely huge, in this case the number of frequent closed patterns is also tend to be very large. In most case, the previously developed closed pattern mining algorithms rely on candidates to check if a newly found frequent pattern is closed or if it can invalidate some already mined closed candidates. Because the set of already mined frequent closed patterns keeps growing during the mining process, not only will it consume more memory, but also lead to inefficiency due to the growing search space for pattern closure checking. In this paper, we proposed FMCSP, a novel algorithm for mining frequent closed sequences. It has improved the drawback of the *candidate maintenance-and-test* paradigm, constructing more compact searching space compare to the previously developed closed pattern mining algorithms. FMCSP adopts a breadth-first method can output the frequent closed patterns online.

References:

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

[2] C. Lucchese, S. Orlando and R. Perego, Fast and Memory Efficient Mining of Frequent Closed Itemsets, IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 1, January 2006.

[3] P. Songram, V. Boonijin and S. Intakosum, Closed Multidimensional Sequential Pattern Mining, Proceeding of the Third Conference on Information Technology: New Generations (ITNG'06).

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

[11] 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.

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

[13] 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.

[14] 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.

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

[16] Fast Accumulation Lattice Algorithm for Mining Sequential Patterns, Proceedings of the 6th WSEAS International Conference on Applied Computer Science (ACOS'07), pp. 230-234, Hangzhou, China, April 15-17, 2007.

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