INTERNATIONAL JOURNAL OF COMPUTERS
Issue 4, Volume 1, 2007

Discover Sequential Patterns in Incremental Database

Nancy P. Lin, Wei-Hua Hao, Hung-Jen Chen, Hao-En, and Chueh, Chung-I Chang

***Abstract*—**The task of sequential pattern mining is to discover **t**he complete set of sequential patterns in a given sequence database with minimum support threshold. But in practice, minimum support some time is defined afterward, or need to be adjusted to discover information that interest to knowledge workers. In the same time, the problem of discover sequential patterns in a incremental database is an essential issue in real world practice of datamining. This paper discusses the issue of maintaining discovered sequential patterns when some information is appended to a sequence database. Many previous works based on Apriori-like approaches are not capable to do so without re-running previously presented algorithms on the whole updated database. We propose a novel algorithm, called DSPID, which takes full advantage of the information obtained from previous mining results to cut down the cost of finding new sequential patterns in an incremental database.

Keywords—Data mining, Sequential patterns, condensed representations, Maximal sequential patterns

I. Introduction

The major issue of data mining in the recent years has been focused on mining sequential patterns in a set of data sequence. Most real world database contains records with time stamp, such as sensor, scientific, monitoring and e-Learning data. The issue of sequential pattern mining was first introduced by Agrawal and Srikant [2] in 1995: *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.*

Manuscript received June 21, 2007. Wei-Hua Hao is with the Department of Computer Science and Information Engineering Tamkang University, 151 Ying-Chuan Road, Tamsui, Taipei, (phone: 886-2-25417002; e-mail:weihua.hao@gmail.com). Revised version September 27, 2007

Wei-Hua Hao is with the Department of Computer Science and Information System of Tamkang University, Taipei, Taiwan. (phone: +886-2-25417002; e-mail:117168@mail.tku.edu.tw).

Nancy Lin is with the Department of Computer Science and Information System of Tamkang University, Taipei, Taiwan. (phone: 886-2-25417002; e-mail:nancylin@mail.tku.edu.tw).

Hung-Jen Chen is with the Department of Computer Science and Information System of Tamkang University, Taipei, Taiwan. (phone: 886-2-25417002; e-mail: chenhj@mail.sju.edu.tw).

Hao-En Chueh is with the Department of Computer Science and Information System of Tamkang University, Taipei, Taiwan. (phone: 886-2-25417002; e-mail: chenhj@mail.sju.edu.tw).

Chung-I Chang is with the Department of Computer Science and Information System of Tamkang University, Taipei, Taiwan. (phone: 886-2-25417002; e-mail: chenhj@mail.sju.edu.tw).

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 has gradually become an important data mining task, with broad applications, including market and customer analysis, web log analysis, intrusion detection system (IDS) and mining XML query access patterns. The revealed information and knowledge are widely used in various applications, including learning status analysis, decision support, and fraud detection. It is one of the most important domains of Data Mining. In these few years many approaches have been proposed to mining sequential patterns. The Sequential pattern mining is now widely used in many areas, such as the analysis of internet intrusion detection, e-Learning sequential patterns, web user behaviors analysis, customer buying behavior analysis and etc.

The major problem in previous works [1], base on apriori-like approach, of this field is that generate too many candidate sequences during the mining process, which increase the requirement of hardware and system runtime. And then closed itemset and maximal itemset, sequence, concept has been introduced [2][3][4][5][6] to mitigate these drawbacks. Apriori employs a bottom-up searching method that enumerates every single frequent sequence. This means in order to generate a frequent sequence of length *l*, it must generate all 2^l of its subsequences since they too must be frequent. This exponentially growing complexity fundamentally restricts Apriori-like algorithms to discover only short patterns. This mining algorithm has a consequence of the following problems: 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. Even though many previous proposed researches have alleviate this drawback via join method generating less candidates, but these redundant is still a major problem that require more memory space to store them and more machine cycle to handle, generate and prune, these unnecessary process. Mining sequential patterns with maximal sequential patterns may largely reduce the number of patterns generated during the process and without losing any information, which is because of it can be used to derive the complete set of sequential patterns. In previous studies [4][5], which have proposed two novel mining algorithms, Fast Accumulated Lattice(FAL) algorithm and Fast Mining Maximal Sequential Patterns(FMMSP), scan sequence database only once, to our knowledge the scan times of data base these algorithms are less than the FP-tree which needs to scan database twice, and further more mining sequential patterns without generating unneeded candidates which are to be pruned in the following mining process. However, in some real world cases the requirement of memory space is critical that demanding a novel algorithm to

minimize, or prevent, generating non-maximal frequent sequences.

Many previous researches gave contributions to mining sequential patterns efficiently of temporal data. Agrawal and Srikant proposed a generalized sequential pattern mining algorithm [6], called GSP, which applied the candidate generation and test, gen and prune, principle. First, it scan database to discover all frequent 1-sequences, sequence length equals to 1. Second, generates candidate of 2-sequences from the sets of frequent 1-sequences. That is, in general, generating candidate (i+1)-sequences from the sets of frequent i-sequences.

To alleviate the drawback of generating huge amount of candidates in the mining process, *Garofalakis* has proposed SPIRIT[7], a Apriori-like algorithm, to generate less candidates via constrains. Jia-Wei Han proposed Prefixspan[8] and Freespan[9] algorithms, which are based on projected databases. These 2 algorithms applied a divide-and-conquer approach, generating many smaller projected databases of the original sequence database, and mining the frequent sequences in each projected databases by discovering participated frequent patterns. In real world data mining application, database is updated from time to time, it is incremental. In this case, many new sequences are newly appended to database, altered the frequent sequential patterns set. The consequence is the previous mining result of frequent sequence has to change with the updated sequences. A common used strategy is rebuilding the frequent sequences from the most up to date database. This is very inefficient especially when dealing with huge amount of data. Obviously, rebuilding from scratch didn't take the advantage of pervious work. Many researches of incremental mining of sequential patterns were developed in recent years. An incremental method SPADE[18] of mining sequential patterns was proposed by *Zaki*. In this paper, equivalent class was introduced to construct sequence lattice in incremental manner. Newly read sequence data from sequence database are updated into the lattice. Other research has presented diverse methods solving the incremental sequence database problem in [23][24]. Many proposed methods on incremental sequence mining have to tackle the problems of dealing with the newly append sequences to the original sequential database to form into previous constructed frequent sequential patterns, and to adjust the sequential patterns with change of minimum support, which is usually change during after, or during, the mining process. In the practice fields, e-commerce and eLearning applications, facing a incremental sequence database is inevitable. How to save mining time with less memory is essential to evaluate sequential patterns algorithm. In this aspect, not to rebuild previous construct sequential patterns is almost an essential part to solve this problem.

In this paper we propose a new algorithm, DSPID, to alleviate this problem via mining frequent sequences in a form of maximal sequential pattern, rather than mining the full set of frequent sequences. The reason why we mine maximal sequential patterns is that they are compact representations of frequent sequential patterns.

In our point of view, the main contributions of this paper are: constructing maximal sequence model without generating redundancies of candidates and non-maximal sequences in the process require less memory space, append new sequence to data model without rebuilding it, smaller searching space and categorized frequent sequential patterns.

The rest of this paper is organized as follows. In section 2, we define the basic definitions and properties of sequential patterns. The algorithm of DSPID and its example are given in section 3 and 4, respectively. Section 5 gives conclusion and future work.

II. PRELIMINARY

The problem can be described as follows: Assume $I=\{i_1, i_2, \ldots, i_n\}$ be a set of all items (or events). An *itemset* is a non-empty set of finite items. A sequence is an ordered list of itemsets. A sequence s is denoted as $s=\langle i_1,\ldots,i_{|s|}\rangle$, where i_i is an itemset, that is , $i_i \quad I$ for $1\le i\le k$. s_i is also defined as an element of sequence, and denoted as $(x_1x_2\ldots x_l)$, where $x_j\in I$ for $1\le j\le l$. In fact, the brackets are usually omitted if $|i_i|=1$. An item can appear at most once in an element of a sequence, but can appear more than one time in different elements of a sequence. The length of a sequence is defined as the number of instances of items in a sequence. A sequence with length l is called an l-sequence. A sequence $x=\langle x_1,x_2,\ldots,x_n\rangle$ is called a subsequence of $y=<y_1,y_2,\ldots,y_m>$ and y a super sequence of x, denoted as x y, y contain x, if there exist integers $1\le j_1 < j_2 < \ldots < j_n\le$m such that . A sequence database D is a set of tuples denoted as $\langle SID, s\rangle$, where SID is a sequence identification number and s is a sequence. Given a k-items sequence s, its support is $supp(s)$ which is defined as the number of transactions in D that including s. Apriori-like algorithm mine all the frequent sequences from 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 I.

Cardinality of s denotes the number of distinct *SID* values in the id-list of sequence database for a sequence s.

The set of maximal sequence is defined as $MS=\{s \mid s\in S \ and \ \neg\exists s'\in S \ such \ that \ s\prec s'\}$.

A sequence X is maximal sequence if there exists no super-sequence Y $\supset$ X, with the same support as X [6].

Let SDB be a sequence database, minimum support is minsup, and NDB be a appended sequence database. PDB is updated sequence database that PDB=SDB+NDB. When original sequence database has changed the algorithm must make use of the previously discovered information to adapt with the change. The idea is constructing a data model representing the original sequence database. The data model has to change with original sequence database without rebuilding the data model. This data model is transform from original sequence database without distortion, DM=f(SDB). When new sequence has been appended to original sequence

database the data model also change dependently. DM'=*f*(SDB)+ *f*(NDB).

III. DSPID Algorithm

DSPID provides a categorized, in frequency, data model represent original sequence database without distortion. With a approach of incremental strategy, sequences of D are read one by one, transformed and load into the data model, Frequent Sequences Set (FSS). Sequence S read from database is compared to the existed sequences in the FSS. Comparison is in descending order in each array, but in ascending order from F1 to higher frequent sequence array. The relationship between S and S_{FSS} are $S \cap S_{FSS} = \phi$, $S \cap S_{FSS} = S$, $S \cap S_{FSS} = S_{FSS}$ or S and S_{FSS} are partially mutual to each other. That is mutual sequence $S_m = S \cap S_{FSS} \neq \phi$ and $S \neq S_{FSS}$. The new sequence S is processed according to the type of relationship. The first case is simple; we just append the new sequence S to the array of F1. In case 2 and 3, the mutual part is upgraded to higher frequent array. In case 4, S is upgraded to higher frequent array. Each frequent array contains maximal sequences only. For example, sequence 〈ABC〉 and 〈C〉 will not coexist in the same array because 〈ABC〉 is the maximal sequence of 〈C〉.

```
==================================
//Input: D
//Output: FSS
Initial 2-dimension array FSS={ F1, F2, …, F1}

Function Upgrade(S){
   move S from allocated array to higher frequent array }
For each sequence in D{
   Read sequence S from sequence database D
   For n= 1 to Top{
        Case S∩ each sequence in Fn = φ :{
           append S to Fn ;
           break;
           }
        Case  S⊂ Fn.sequence  :{
           Upgrade(S);
           mark S;
           }
        Case  S⊃ Fn.sequence:{
           Upgrade(Fn.sequence);
           Mark Fn.sequence
           }
        Case  S∩ Fn.sequence=Smutual :{
           Upgrade(Smutual);
           S= Smutual;
           }
   }
}
==================================
```

Fig. 1 DSPID algorithm

IV. Example

We will demonstrate how the DSPID is capable to build a data model representing original sequence database, minimize the searching space and accelerate runtime with example. In Figure. 2 is a simple sequence database, D. SID represents Student Identifier. The itemset include A, B, C, D and E.

SID	Sequence
1	ACD
2	ABCE
3	BCE
4	BE
5	ABCDE

(D)

F1	F2	F3	F4

Empty (FSS)

Fig.2 An original sequence database D and data model FSS

First, we consider the construction of data model with DSPID algorithm. When read in the first sequence 〈ACD〉 from database D. Figure.3 shows the new sequence is allocated to frequent-1 array, F1. And new sequence is compare with all sequences in F1. Since there is no other sequence in F1 the algorithm stops the comparison process.

F1	F2	F3	F4
ACD			

Fig. 3 FSS containing 〈ACD〉

In Figure.4, continue to read in 〈ABCE〉. First, the new sequence is allocated to F1, next to previous sequence. Compare sequence 〈ABCE〉 with 〈ACD〉, the mutual sequence of 〈ACD〉 and 〈ABCE〉 is 〈AC〉 which will be upgraded to higher frequency array F2. As shown in Fig 4.

F1	F2	F3	F4
ACD	AC		
ABCE			

Fig. 4 FSS with 〈ACD〉 and 〈ABCE〉 and their mutual sequence 〈AC〉

The next sequence reading from D is 〈BCE〉. Compare 〈BCE〉 with F1, found out that sequence 〈BCE〉 is contained by 〈

ABCE〉, so 〈BCE〉 is upgraded to F2. Sequence 〈BCE〉 is a new sequence to F2. The mutual sequence of 〈BCE〉 and 〈AC〉 is 〈C〉 which will be upgraded to F3. As shown in Figure. 5. The sequence 〈BCE 〉 has been upgraded from F1 to F2. The upgrade algorithm will leaves a marked, under line, sequence of 〈BCE〉 in F1.

F1	F2	F3	
ACD	AC	C	
ABCE			
BCE			

Allocate 〈BCE〉 to F1

F1	F2	F3	
ACD	AC		
ABCE	BCE		
BCE			

Upgrade 〈BCE〉 to F2

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE		
BCE			

Fig. 5 example of upgrading mutual sequence

Next sequence read from database D 〈BE〉 is contained by 〈ABCE〉 of F1, so 〈BE〉 is upgraded to F2. Compare 〈BE〉 with sequences in F2. Obviously, 〈BE〉 is contained by 〈BCE〉 so 〈BE〉 is upgraded to F3. See Figure 6.

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE		
~~BCE~~			
BE			

Allocate 〈BE〉 to F1

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE		
BCE	BE		
BE			

Upgrade 〈BE〉 to F2

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE	BE	
BCE	BE		
BE			

Fig. 6 upgrading 〈BE〉

Last sequence 〈ABCDE〉 contains all sequences in F1, so 〈ACD〉〈ABCE〉 are upgraded to higher frequency array F2. As shown in Fig. 7.

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE	BE	
BCE	BE		
BE			
ABCDE			

Allocate 〈ABCDE〉 to F1

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE	BE	
BCE	BE		
BE	ACD		
ABCDE			

Upgrade 〈ACD〉 to F2

F1	F2	F3	F4
ACD	AC	C	
ABCE	BCE	BE	
BCE	BE	AC	
BE	ACD		
ABCDE			

Upgrade 〈AC〉 to F3

F1	F2	F3	F4
ACD	AC	C	C
ABCE	BCE	BE	
BCE	BE	AC	
BE	ACD		
ABCDE			

Upgrade 〈C〉 to F4

F1	F2	F3	F4
ACD	AC	C	C
ABCE	BCE	BE	
BCE	BE	AC	
BE	ACD		
ABCDE	ABCE		

Upgrade 〈ABCE〉 from F1 to F2

F1	F2	F3	F4
ACD	AC	C	C
ABCE	BCE	BE	
BCE	BE	AC	
BE	ACD	BCE	

ABCDE	ABCE		

Upgrade 〈BCE〉 from F2 to F3

F1	F2	F3	F4
ACD	AC	C	C
ABCE	BCE	BE	BE
BCE	BE	AC	
BE	ACD	BCE	
ABCDE	ABCE		

Upgrade 〈BE〉 from F3 to F4

F1	F2	F3	F4
			C
			BE
		AC	
	ACD	BCE	
ABCDE	ABCE		

Hide marked sequences in FSS

F1	F2	F3	F4
ABCDE	ACD	AC	C
	ABCE	BCE	BE

Display FSS in compact format

Fig.6 Complete FSS of D

If we set the threshold to 3 then the frequent sequences are given by DSPID algorithm immediately via the FSS table. The frequent sequences are 〈AC:3〉 〈BCE:3〉 〈C:4〉 and 〈BE:4〉.

V. CONCLUSION

Compare to previous works the advantages are: No candidates were generated during DSPID mining process that saves a lot of memory unit both in hard disk and RAM. Search space is no longer an issue to DSPID algorithm because the output is a categorized maximal frequency sequence arrays that can be set to any threshold or minsup. This gives knowledge worker privilege to adjust the threshold according their domain knowledge.

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.

Apriori involves a phase for finding patterns called *frequent itemsets*. A frequent itemset is a set of items appearing together in a number of database records meeting a user-specified threshold. Apriori employs a bottom-up search that enumerates every single frequent itemset. This implies in order to produce a frequent itemset of length; it must produce all of its subsets since they too must be frequent. This exponential complexity fundamentally restricts Apriori-like algorithms to discovering only short 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 large, in this case the number of frequent maximal sequential patterns is also tend to be very large. In this paper, we proposed DSPID, a novel algorithm for mining frequent maximal sequential sequences. It has improved the drawback of the *candidate maintenance-and-test* paradigm, constructing more compact searching space compare to the previously developed maximal pattern mining algorithms. DSPID adopts a breadth-first method can output the frequent maximal 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] Nancy P. Lin, Wei-Hua Hao and Hung-Jen Chen, 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.

[5] Nancy P. Lin, Wei-Hua Hao, Hung-Jen Chen, Hao-En Chueh and Chung-I Chang, Fast Mining Sequential Patterns, Proceedings of the 7th WSEAS International Conference on Simulation, Modeling and Optimization Applied Computer Science (SMO '07), pp. 405-408,Beijing ,China ,September 15-17, 2007.

[6] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," Proc. Fifth Int'l Conf. Extending Database Technology (EDBT '96), pp. 3-17, Mar. 1996.

[7] M. Garofalakis, R. Rastogi, K. Shim, "SPIRIT: Sequential pattern mining with regular expression constraints," Proceedings of the 25th International Conference on Very Large Databases (VLDB'99), pp. 223-234, 1999.

[8] 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.

[9] 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.

[10] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, Discovering frequent closed itemsets for association rules," In Proc. Seventh Int. Conf. Database Theory (ICDT '99), pp. 398-416, Jan. 1999.

[11] R.J. Bayardo Jr., Efficiently mining long patterns from databases., Proceedings of the international conference on Management of data (SIGMOD'98), 1998.

[12] 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.

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

[20] 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.

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

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

[23] F. Masseglia, P. Poncelet, M. Teisseire, "Incremental mining of sequential patterns in large databases," Actes des Jouenes Bases de Donnes Avances (BDA'00), Blois, France, 1999.

[24] Weimin Ouyang, Qingsheng Cai, "An incremental updating techniques for discovering generalized sequential patterns," Journal of Software, Vol. 9, No. 10, pp. 778-780, 1998.