IncSPADE: An Incremental Sequential Pattern Mining Algorithm Based on SPADE Property

Omer Adam, Zailani Abdullah, Amir Ngah, Kasypi Mokhtar, Wan Muhamad Amir Wan Ahmad, Tutut Herawan, Noraziah Ahmad, Mustafa Mat Deris, Abdul Razak Hamdan and Jemal H. Abawajy

Abstract In this paper we propose Incremental Sequential PAttern Discovery using Equivalence classes (IncSPADE) algorithm to mine the dynamic database without the requirement of re-scanning the database again. In order to evaluate this algorithm, we conducted the experiments against three different artificial datasets. The result shows that IncSPADE outperformed the benchmarked algorithm called SPADE up to 20%.

Keywords Sequential pattern · Incremental · Updatable · Database

O. Adam (✉) · Z. Abdullah · A. Ngah
School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Malaysia
e-mail: gsk2211@pps.umt.edu.my

Z. Abdullah
e-mail: zailania@umt.edu.my

A. Ngah
e-mail: amirnma@umt.edu.my

K. Mokhtar
School of Maritime Business and Management, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Malaysia
e-mail: kasypi@umt.edu.my

W.M.A.W. Ahmad
School of Dental Sciences, Universiti Sains Malaysia, 16150 Kubang Kerian, Kelantan, Malaysia
e-mail: wmamir@usm.my

T. Herawan
Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
e-mail: tutut@um.edu.my

P.J. Soh et al. (eds.), *Advances in Machine Learning and Signal Processing*, Lecture Notes in Electrical Engineering 387, DOI 10.1007/978-3-319-32213-1_8

1 Introduction

The main goal of data mining is to discover the hidden patterns from a large amount of data [1]. There are many data mining problems have been introduced and one of them is sequential pattern mining [2]. Sequential pattern mining is an evolving data mining problem in various domain applications such as in marketing to find customer purchase patterns, web access patterns, DNA sequence analysis [3], etc. Nowadays, sequential pattern mining has received a great attention since it was first proposed by Agrawal and Srikant in 1995 [3]. The problem of sequential patterns mining is given a set of sequence where each sequence contains set of elements and each element consist of set of items, given a user specified minimum support threshold, and finally discover all frequent subsequence [2]. Up to date, there are many algorithms have been proposed for sequential pattern mining based on the property of Apriori algorithm. Apriori algorithm was introduced by [3] to find frequents sequence from given dataset through recursive and iterative procedures.

Apriori-based and Pattern growth [4] are the most common sequential pattern approaches. Pattern growth algorithms appeared in the early 2000s to solve the problem of generating candidates and test. It has few proposed algorithms compared to Apriori-based approach, the most common pattern growth algorithms are IncSpan [5], FreeSpan [6] and PrefixSpan [4]. Generally, Apriori-based approach algorithms can be categorized into horizontal and vertical data formats. Examples of horizontal data format algorithms are Apriori [3] and GSP [7] and vertical data format algorithms are SPADE [8] and SPAM [9]. The vertical format provides the advantage in generating the patterns and calculating their supports without performing costly I/O during database scans [10].

Although vertical algorithms have better performance in dense dataset as compared to horizontal algorithms, yet this performance is getting slow when it comes

N. Ahmad
Universiti Malaysia Pahang, Malaysia, 26300 Kuantan, Pahang, Malaysia
e-mail: noraziah@ump.edu.my

M.M. Deris
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia
e-mail: mmustafa@uthm.edu.my

A.R. Hamdan
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
e-mail: arh@ftsm.ukm.my

J.H. Abawajy
School of Information Technology, Deakin University, Burwood Campus/221, Waurn Ponds, VIC 3216, Australia
e-mail: jemal.abawajy@deakin.edu.au

to a dynamic database [11]. A dynamic database is a database that is frequently updated and increased in term of number of records [12]. This update can occur in many different ways, such as APPEND to existing record (existing customer buys new items), INSERT new record (new customers buy items) and DELETE record (current records have been removed from the database) [5]. The idea of incremental sequential patterns is giving an updated database find the current frequent patterns [13, 14]. As a result to mine the latest patterns from updated database, most of the current algorithms have to rescan back from entire updated database [15]. Therefore, in this paper we propose an incremental sequential pattern algorithm IncSPADE is based on SPADE propriety with fixable data structure to mine updated database without having to start the mining process from the scratch once the database is updated. Experiments result shows that IncSPADE outperform SPADE by 20% in term of performance in most cases. IncSPADE algorithm constructs a tree map data structure (FINFT) stand for frequent infrequent tree map, that contains all frequent and infrequent sequences generated during the first time the algorithm start. Maintain such tree makes it easy to IncSPADE to mine updated database without rescanning the whole database.

The rest of the paper organized as follow: in Sect. 2 discuss related work in the field, Sect. 3 describe the methodology, Sect. 4 explains the obtained result and discussion and finally in Sect. 5 draw a conclusion and our further research direction

2 Related Works

Several algorithms have been proposed for mining incremental association rules, however only a little attention has been given to sequential pattern mining [16]. Most of the proposed solutions to mine incremental sequential pattern are Apriori-based and pattern growth approaches. SPADE was first proposed by [8]. It uses Eclat [17] approach which was proposed by the same author for mining frequents items. The key feature of SPADE is the layout of the database in a vertical id list format. SPADE breaks the search space into sub-lattices tree-equivalence classes—that can be processed independently in main memory [10]. It scans the database three time only or just once with some data preprocessing [8]. First SPADE scans the database to construct frequent 1-sequence. After frequent 1-sequence prefix tree equivalence classes have been set with root null, SPADE starts enumeration process to compute frequent k- sequence. The process of generating k-sequence done by using frequent k-1-sequence equivalence classes as prefix and generate k-sequence with simple join operation from k-1-sequence [18]. During sequence generation, SPADE prunes the items against minim support sub and if sequence TIDs less than minim support SPADE will not add to prefix tree and thus reducing the number of next generated sequences [19]. SPADE outperforms Apriori-all and GSP by an order of magnitude in most cases sequence [8].

Table 1 Sequence database

CID	TID	Items
1	10	A B
	20	B
	30	A B
2	20	A C
	30	A B C
	50	B
3	10	A
	30	B
	40	A
4	30	A B
	40	A
	50	B

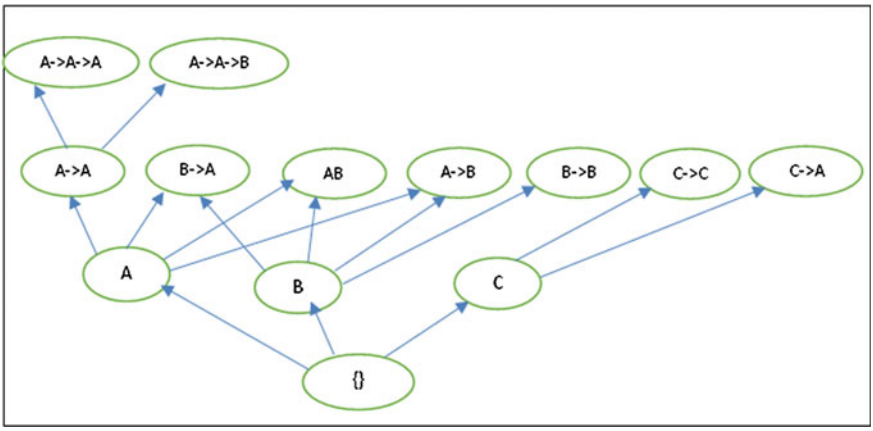

Fig. 1 Example of equivalence classes

Table 1 shows a sequence database and Fig. 1 shows the lattice tree constructed by SPADE for this database. Nodes marked in gray are infrequent sequences.

FASTUP algorithm [20], is one of the earliest work in incremental sequential pattern mining. FASTUP is an enhancement of GSP algorithm [7], FASTUP takes into account the previous result generated before start generating and examine candidates using generate and prune technique. The idea of having knowledge of previous result is to take advantages of the information related to sequence threshold to generated candidates, therefore, FASTUP avoids the issue of generating sequence depend on their support [16].

Comprehensive Incremental mining algorithm of Closed sequence (CISpan) proposed by [7]. It's an enhancement of the Clospan [21] algorithm [22]. The advantage of CISpan is the ability to remove sequence from the database in addition

to database INSERTION. CISpan uses divide and conquer approach that makes INSERTION and DELETION operation are in depended from each other [23]. In INSERTION, the algorithm constructs an incremental lattice prefix tree to keep all frequents sequence that found in the updated database. However, removal of the sequence is done by updating the original database lattice tree. In the process of INSERTION CISpan only build a tree lattice for the newly inserted sequence, then CISpan merges the original prefix lattice with the incremental lattice result into updated frequent sequences [4].

3 Methodology

Up to our knowledge, there is not many works have been done in incremental sequential data mining. This section highlights some basic definitions and the proposed algorithm. IncSPADE is based on SPADE properties. IncSPADE is able to handle the updated database without reexamining the database from the scratch. There are two main phases of the IncSPADE, in the first phase IncSPADE behave just like SPADE. The second phase after the database has been updated; IncSPADE runs to generate the new frequents sequences.

3.1 Preliminary Definitions

Definition 1 (*sequence database*) A sequence is an ordered list of itemsets. Let *i* be $<i_1\ i_2 \ldots i_m>$ where i_j is an item, we donate a sequence *s* by $<s_1\ s_2 \ldots s_n>$ where s_j is an itemset.

Definition 4 (*support*) The *support* of a sequence *sa* in a sequence database *SDB* is defined as the number of sequences $s \in SDB$ such that $sa \sqsubseteq s$ and is denoted by *sup SDB* (*sa*).

Definition 3 (*prefix*) A sequence $sa = A_1, A_2, \ldots, A_n$ is a *prefix* of a sequence $sb = <B_1, B_2, \ldots, B_m>$, $\forall n < m$, iff $A_1 = B_1, A_2 = B_2, \ldots, A_{n-1} = B_{n-1}$ and the first $|A_n|$ items of B_n according to $\succ$. *lex* are equal to A_n.

Definition 5 (*sequential pattern mining*) Let *MinSupp* be a threshold set by the user and *SDB* be a sequence database. A sequence *s* is a *sequential pattern* and is considered *frequent* iff $supSDB(s) \geq MinSupp$.

Definition 2 (*Horizontal data format*) A sequence database in horizontal format is a database where each entry is a sequence. The horizontal data format was first introduced by Agrawal and Srikant upon proposing Apriori algorithm, its represents the items categorized into particular transactions. Table 2 shows a sample of horizontal sequence database.

Table 2 A sample of horizontal sequence database

SID	Sequence
1	<{A,B},{B},{A,B}>
2	<{A,C},{A,B,C},{B}>
3	<{A},{B},{A}>
4	<{A,B},{A},{B}>

Table 3 Vertical sequence database

A		B		C	
SID	Itemset	SID	Itemset	SID	Itemset
1	1,3	1	1,2,3	1	
2	1,2	2	2,3	2	1,2
3	1,3	3	2	3	
4	1,2	4	1,3	4	

Definition 3 (*Vertical data format*) A sequence database in vertical format is a database where each entry represents an item and indicates the list of sequences where the item appears and the position(s) where it appears. Table 3 shows the same as Table 1 database in the vertical format.

Definition 4 (*APPEND record*) An APPEND sequence database $DB\sim$ is a record that been appended to an existing record in original database, lets *DB* be original database and lets S=<s_1, s_2, s_3, … s_n> a sequence items in D, an updated database $D^{\sim}$ with S ~ <i_1, i_2, i_3 … i_n> final database $DB_F = DB \cap DB\sim$. Appended database has same numbers or fewer transactions as the original database.

Definition 5 (*INSERTION record*) An INSERTION sequence database $DB\sim$ is a records that been inserted to existing database, lets *DB* be original database and lets S=<s_1, s_2, s_3, … s_n> S= a sequence items in *DB*, an inserted database $DB\sim$ with S ~ <i_1, i_2, i_3 … i_n> final database $DB_F = DB \cup DB\sim$. INSERTION database has same or more transaction as the original database. Table 4 shows example of sequence database and its update in darker background that depict APPEND and INSERTION operations.

3.2 Incremental Sequential Mining

Let $|DB|$ be the number of data sequences in the original database and *MinSupp* is the minimum support. After the update of the original database $DB\sim$ a new records are added to original *DB*. The new records will be categorized by their *cid* in *DB*, each record will be merged with its corresponding *cid*. Thus final updated database $DB_F = DB \cup DB\sim$ where $DB\sim$ is the inserted dataset. In case of the *cid* record does not match with existing *cids* this records will be added as new *cid* and consider as INSERTION operation. Unlike APPEND database INSERTION requires overall

Table 4 Sequence database and its update

CID	TID	Items
1	10	A B
	20	B
	30	A B
	40	**C**
2	20	A C
	30	A B C
	50	B
	60	**A**
3	10	A
	30	B
	40	A
4	30	A B
	40	A
	50	B
	10	**AB**

changes in the frequent sequences tree structure, where some sequences might become infrequent due to change of the *MinSupp*.

IncSPADE is based on SPADE algorithm, its take an update database $DB\sim$ and run it against the previous obtained frequent and infrequent sequences tree, IncSPADE does not need to run the database from scratch again, it is only built prefix lattice tree structure for the newly appended sequences which did not appear previously on the original database. Using this idea, IncSPADE reduce time and memory space needed to mind the original and the updated database. Figures 2 and 3 presents IncSPADE model and IncSPADE algorithm, respectively.

1. **Scan database**

First step IncSPADE scans the original database to count the items and generate 1-freqnet sequence itemset.

2. **Create equivalence classes and enumerate**

After 1–frequent itemset generated, IncSPADE constructs a tree lattice search with root null and start enumeration process to generate k-frequents itemset from the prefix equivalence class k-1 frequent itemset.

3. **Frequent and none frequents sequence**

Once IncSPADE finished enumeration over the equivalence classes and prune sequence items against MinSupp, frequent and none frequent sequence.

4. **Scan updated DB**

This is a critical phase where IncSPADE scans the updated database to count the new sequence items. During this phase IncSPADE checks updated database size, if

```
1     DB = Original database
2.    Start
3     Scan DB to find frequent and infrequent items
4     Generate frequents sequence
5     Construct frequent and infrequent tree map
6     UDB = updated database
7     Scan UDB
8.    If checkItem(item)
9.     addToSearchTree(item);
10    Else
11.     CheckfrequentsItems(item)
12.       If isFrequent(item)
13.         updateItemTdis(item)
14.       else checkInfrequents(item)
15.           If ItemInfrequent(item)
16.             CheckItemTids(item)
17.               If (tisd > miusup)
18.                 AddedItemToFrequent(item)
19                  RemoveItemFromInfrequent(item)
20.           else addToSearchTree(item)
21            Endif
22          Endif
23        Endif
24    endif
25.   forall New items do
26     GenerateCandidate()
27.     If candidate tdis > minsup
28.      Add to frequent
29.     Else added to infrequent
30      Endif
31    Endfor
32    Output = Generate frequents sequence
33.   End
```

Fig. 2 IncSPADE algorithm

size is less or equal the original database, IncSPADE consider this update as APPEND else it will be considered as both APPEND and INSERT, MinSupp has to be recalculated then exams the existing frequent sequences against the new MinSupp.

5. **Enumerate and find 1-freqent**

After new items counted and process type identified, tree search will be constructed for the new item. IncSPADE enumerates to generating equivalence classes and prune the new items against frequents and none frequents itemset.

6. **Frequent and none frequents sequence**

At the end of enumerating process, IncSPADE come out will the new frequent and none frequents itemset which reflects the original and update databases with respect to MinSupp.

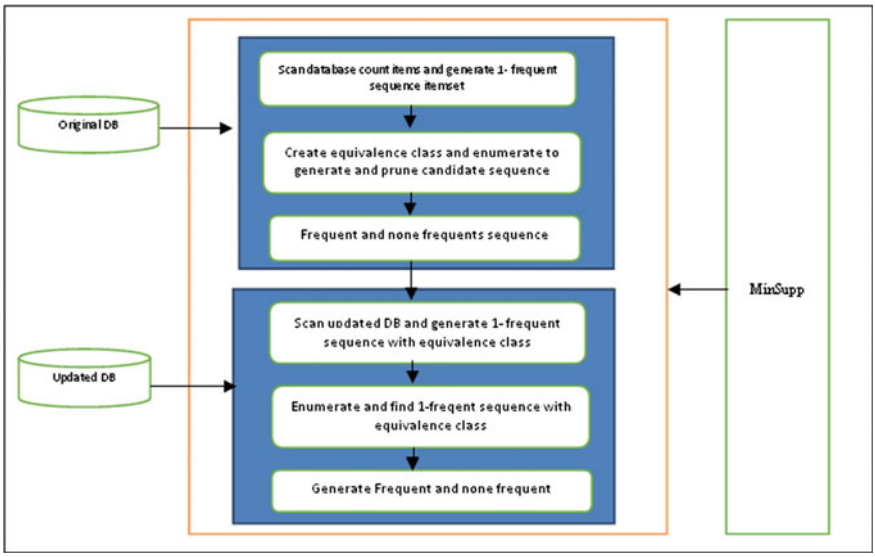

Fig. 3 IncSPADE model

4 Result and Discussion

The experiment was carried out on Intel® Core™ i3-3210 M CPU at 2.50 GHz speed with 1.00 GB RAM, running on Window 7 Home Premium 64bits. IncSPADE algorithm has been developed using Netbeans IDE 8 with Java JDK 1.8 and Java as a programming language.

Three artificial datasets was generated using our dataset generator. The datasets scenarios as follow: first dataset (DATA1) contains 20 items and 10 items per record, second dataset (DATA2) contains 30 items and 20 items per record. We split into at least two parts to stimulate database. The complete two datasets was run on SPADE algorithm with different MinSupp then continued to run the split datasets in IncSPADE. The datasets was split into two equal parts of 25,000 records for IncSPADE. The result shows that IncSPADE outperform SPADE in most cases by 20%. This is because IncSPADE used the previous knowledge to capture the frequent and none frequent sequences before determining the new frequent items. It helps IncSPADE to reduce the time complexity of generating and searching the new frequents items. Table 5 shows the datasets properties. Figures 4 and 5 depict the performance comparison between IncSPADE and SPADE for each dataset.

Table 5 Artificial datasets properties

Dataset	Number of records	Number of items	Items per record
Data 1	500,000	20	10
Data 2	500,000	30	20

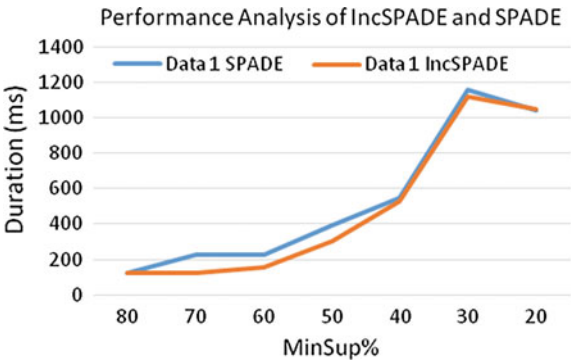

Fig. 4 Performance analysis of IncSPADE and SPAD

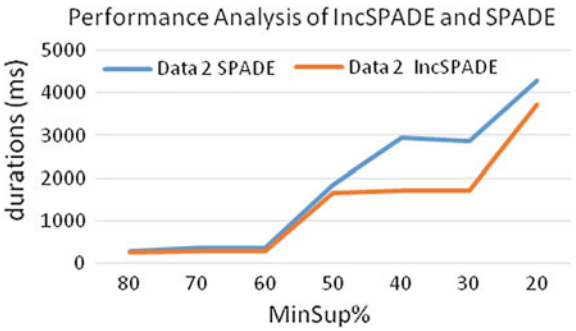

Fig. 5 Performance analysis of IncSPADE and SPADE

5 Conclusion

In this paper, we proposed IncSPADE algorithm to solve the issue of mining dynamic database without re-mine the database from scratch each time its updated. The result shows that IncSPADE was able to mine the updated database. In the near future, the algorithm will be enhanced and optimized to deal with the issue of constructing the frequent and infrequent tree.

References

1. Kumar V, Anupama C (2012) Mining association rules in student's assessment data. Int J Comput Sci Issues 9(5):211–216
2. Vishal SM (2014) A survey on sequential pattern mining algorithm. Int J Comput Sci Inf Technol 5(2):2486–2492
3. Agrawal R, Ramakrishnan S (1995) Mining sequential patterns. In: Proceedings of the 11th international conference on data engineering, pp 3–14
4. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2001) Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceeding of 17th international conference on data engineering
5. Hong C, Yan X, Han J (2004) IncSpan: incremental mining of sequential patterns in large database. In: Proceedings of the 10th ACM SIGKDD international conference on knowledge discovery and data mining, pp 527–532
6. Han J, Dong G, Mortazavi-Asl B, Chen Q, Dayal U, Hsu M-C (2000) FreeSpan: frequent pattern-projected sequential pattern mining. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining, pp 355–359
7. Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. Adv Database Technol LNCS 1057:1–17
8. Zaki M (2001) SPADE: an efficient algorithm for mining frequent sequences. Mach Learn 42:31–60
9. Jay A, Gehrke J, Yiu T, Flannick J (2002) Sequential pattern mining using a bitmap representation. In: Proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining, pp 429–435
10. Fournier-Viger P, Gomariz A, Campos M, Thomas R (2014) Fast vertical mining of sequential patterns using co-occurrence information. LNAI 8443:40–52
11. Parthasarathy S, Zaki MJ, Ogihara M, Dwarkadas S (2002) Sequence mining in dynamic and interactive environments. Knowl Discov Bus Inf Syst 600:377–396
12. Lin MY, Lee SY (2004) Incremental update on sequential patterns in large databases by implicit merging and efficient counting. Inf Syst 29(5):385–404
13. Gupta M, Han J (2012) Approaches for pattern discovery using sequential data mining. Pattern discovery using sequence data mining: applications and studies, pp 137–154
14. Ezeife CI, Liu Y (2009) Fast incremental mining of web sequential patterns with PLWAP tree. Data Min Knowl Discov 19(3):376–416
15. Ezeife CI, Chen M (2004) Incremental mining of web sequential patterns using PLWAP tree on tolerance MinSupport. In: Proceeding of international database engineering and applications symposium, 2004, pp 465–469
16. Florent M, Poncelet P, Teisseire M (2003) Incremental mining of sequential patterns in large databases. Data Knowl Eng 46(1):97–121
17. Zaki M (2000) Scalable algorithms for association mining. Knowl Data Eng IEEE Trans 12 (3):372–390
18. Leleu M et al (2003) GO-SPADE: mining sequential patterns over datasets with consecutive repetitions. LNAI 2734:293–306
19. Mooney CH, John F (2013) Sequential pattern mining approaches and algorithms. ACM Comput Surv 45(2):2–39
20. Lin M-Y, Lee S-Y (2004) Incremental update on sequential patterns in large databases by implicit merging and efficient counting. Inf Syst 29(5):385–404
21. Wang J, Han J (2004) BIDE: efficient mining of frequent closed sequences. In: Proceedings of 20th international conference on data engineering, pp 79–90
22. He H, Wang D, Chen G, Zhang W (2014) An alert correlation analysis oriented incremental mining algorithm of closed sequential patterns with gap constraints. Int J Appl Math Inf Sci 8 (1L):41–46

23. Mallick B, Garg D, Grover PS (2013) Incremental mining of sequential patterns: progress and challenges. Intel Data Anal 17(3):507–530
24. Yuan D, Lee K, Cheng H, Krishna G, Li Z, Ma X, Zhou Y, Han J (2006) CISpan: (2008) comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining. In: Proceeding of SDM'2008, pp 84–95