Journal of Science and Technology 54 (3A) (2016) 106-114

APPLYING THE ATTRIBUTED PREFIX TREE FOR MINING CLOSED SEQUENTIAL PATTERNS

Pham Thi Thiet[*], Vo Thi Thanh Van

Faculty of Information Technology, Industrial University of Ho Chi Minh City, 12 Nguyen Van Bao Street, Ward 4, Go Vap District, HoChiMinh City

Email: *phamthithiet@iuh.edu.vn*

Received: 4 May 2016; Accepted for publication: 28 July 2016

ABSTRACT

Mining closed sequential patterns is one of important tasks in data mining. It is proposed to resolve difficult problems in mining sequential pattern such as mining long frequent sequences that contain a combinatorial number of frequent subsequences or using very low support thresholds to mine sequential patterns is usually both time- and memory-consuming. This paper applies the characteristics of closed sequential patterns and sequence extensions into the prefix tree structure to mine closed sequential patterns from the sequence database. The paper uses the parent–child relationship on prefix tree structure and each node on prefix tree is also added fields to determine whether that is a closed sequential pattern or not. Experimental results show that the number of sequential patterns is reduced significantly.

Keywords: sequential pattern, closed sequential pattern, prefix tree, sequence database.

1. INTRODUCTION

Sequential pattern mining, since it was first introduced by Agrawal [1], has played an important role in data mining tasks with broad applications including market and customer analysis, web log analysis, pattern discovery in protein sequences, and mining XML query access patterns for caching and so on. The sequential pattern mining algorithms proposed so far have a good performance in databases with short frequent sequences [2, 3, 9 - 10, 16]. However, when mining long frequent sequences that contain a combinatorial number of frequent subsequences, such a mining will generate an explosive number of frequent subsequences for long patterns, or when using very low support thresholds to mine sequential patterns, which is prohibitively expensive in both time and space. So, the performance of such algorithms often degrades dramatically. To overcome this difficultly, the problem for mining closed sequential patterns have also been proposed. A sequence S is called closed if there exists no supersequence of S with the same support in the database. For example, sequence pattern $\langle(AB)(B)(BC)\rangle$ is a closed sequence pattern of sequence patterns $\langle(B)(BC)\rangle$ and $\langle(AB)(B)\rangle$ if they have the same support.

Several studies have been recently proposed to mine closed sequential patterns [4, 11 - 12, 14 - 15]. But these algorithms used the corresponding projected databases of frequent subsequences to find closed sequences. It consumes much time to construct projected databases of frequent subsequences for a set of sequence. In this paper, we propose an efficient algorithm for directly finding closed sequential patterns at the generating sequential patterns process, which is based on the combination of the parent–child relationship and its propertied on prefix tree structure. On the prefix tree in our approach, each node stores a sequential pattern and its corresponding support value. Besides, it will be added one field to consider whether this node is a closed sequential pattern (*IsCSP*). Based on the *IsCSP* field added to each node, the algorithm easily determines if a node is a closed sequential pattern so the mining time is reduced significantly. This algorithm also uses join operations over the prime block encoding approach of the prime factorization theory to represent candidate sequences and determine the frequency for each candidate. The experimental results showed that the performance for mining closed sequential patterns in this paper is much better.

The rest of this paper is organized as follows. Section 2 reviews some works related to mine closed sequential patterns. Section 3 presents some problem definitions related to sequential patterns/closed sequential pattern and prefix tree. The algorithm for mining closed sequential patterns is discussed in Section 4. Section 5 presents the experimental results, and conclusion and future work are presented in Section 6.

2. RELATED WORK

Mining sequential patterns with closed patterns may significantly reduce the number of patterns generated in the mining process without losing any information because it can be used to derive the complete set of sequential patterns; so, the number of closed sequential patterns is usually fewer than the number of sequential patterns. Several studies have been recently proposed to mine closed sequential patterns [4, 11 - 12, 14 - 15]. The CloSpan algorithm [15] has been proposed. Like most of the frequent closed itemset mining algorithms CLOSET [6] and CHARM [17], CloSpan algorithm used the candidate maintenance and test approach. It needs to maintain the set of already mined closed sequence candidates for doing the backward subpattern and backward superpattern check to verify if a newly found frequent sequence is promising to be closed or not. Because CloSpan needs to maintain the set of historical closed sequence candidates, when there are many frequent closed sequences, it will consume much memory and lead to huge search space for pattern closure checking. BIDE [14] is another faster closed sequence mining algorithm. Different from CloSpan, it used a novel sequence closure checking scheme called BI-Directional Extension, and pruned the search space more by using the BackScan pruning method and the ScanSkip optimization technique to directly get the complete set of the frequent closed sequence patterns without candidate maintenance. Thus, in most cases, BIDE is more efficient than CloSpan, especially when a database is dense or the minimum support value is low. But to implement the closure check, the BIDE algorithm spends a lot of time on scanning the pseudo-projected database repeatedly to verify the existence of extension of position with a prefix sequence, which costs much time in the mining process. To reduce the time consumed on scanning the pseudo-projected database for verifying in the BIDE algorithm, the FCSM-PD algorithm was proposed by Huang et al. [4]; the positional data was used to reserve the position information of items in the data sequences. In the pattern growth process, the extension of position with a prefix sequence is checked directly and all the position information of the new prefix sequences will be recorded. The FCSM-PD algorithm must store all the position information of a prefix sequence in the process of pattern growth in advance; so it consumes more memory in this algorithm. In 2013, Pham et al. proposed an algorithm called CloGen algorithm [7]. It used the parent–child relationship on prefix tree structure to find closed sequences and their corresponding generators at the same time. The result of CloGen algorithm has been used to generate non-redundant sequential rules [8]. This paper is using the parent–child relationship on prefix tree structure to find only closed sequences and it's result can be used to mine sequential rules.

3. PROBLEM DEFINITIONS

A sequence database *SD* is a set of sequences $S = \{s_1, s_2, ..., s_n\}$ and a set of items I = $\{i_1, i_2, ..., i_n\}$, where each sequence s_x is an ordered list of itemsets $s_x = \{x_1, x_2, ..., x_n\}$, and s_1 occurs before s_2, which occurs before s_3, and so on, such that $x_1, x_2, ..., x_n \subseteq I$. The size of a sequence is the number of itemsets in the sequence. The length of a sequence is the number of instances of items in the sequence. A sequence with length l is called an l-pattern sequence. A sequence with size *k* is called a *k*-sequence.

Given two sequences $\alpha = \langle a_1\ a_2\ ...\ a_n\rangle$ and $\beta = \langle b_1\ b_2\ ...\ b_m\rangle$ *(where a_i, b_i are itemsets)*, sequence α is called a subsequence of β and β is a supersequence of α, denoted as $\alpha \subseteq \beta$, if there exist integers $1 \le j_1 < j_2 < ... < j_n \le m\ (n \le m)$ such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}$, . . . , $a_n \subseteq b_{jn}$. For example, if $\alpha = \langle(ab), d\rangle$ and $\beta = \langle(abc), (de)\rangle$, where *a, b, c, d*, and *e* are items, then α is a subsequence of β and β is a supersequence of α. The support of a sequence α (denoted by $Sup(\alpha)$) in a sequence database is the number of sequences in the database containing α. Sequence α is a frequent sequence in sequence database *SD*, if $Sup(\alpha) \ge minSup$ where *minSup* is the minimum support threshold defined by user. A frequent sequence is called a sequential pattern. A sequential pattern α is called a closed sequential pattern if and only if $\neg\exists\ \beta$ such that $\alpha \subseteq \beta$ (i.e., β contains α) and $Sup(\alpha) = Sup(\beta)$. Considering the sequence database in table 1 and *minSup* = 50%, sequences $\langle(AB)\rangle$, $\langle(B)(BC)\rangle$ and $\langle(B)(B)(BC)\rangle$ are sequential patterns because their support are greater and equal than *minSup* ($\langle(AB)\rangle$ has the support value 4, $\langle(B)(BC)\rangle$ has the support value 3, and $\langle(B)(B)(BC)\rangle$ has the support value 3). Sequential pattern $\langle(B)(B)(BC)\rangle$ is a closed sequential pattern $\langle(B)(BC)\rangle$

Sequence α is a prefix of β if and only if $a_i = b_i$ for all $1 \le j \le n$. After eliminating the prefix part α of sequence β, the remainder of β is a postfix of β. From the above definition, we know that a sequence of size *k* has *(k-1)* prefixes. For example, a sequence $\langle(A)(BC)(D)\rangle$ has 2 prefixes: $\langle(A)\rangle$ and $\langle(A)(BC)\rangle$. Therefore, $\langle(BC)(D)\rangle$ is the postfix for prefix $\langle(A)\rangle$, and $\langle(D)\rangle$ is the postfix for prefix $\langle(A)(BC)\rangle$.

A prefix tree is similar to a lexicographic tree, which starts from the tree root at level 0. In this paper, the prefix tree is started at the root with a null sequence $\varnothing$. Each child node stores a sequential pattern, its support value, one field to consider whether this node is a closed sequential pattern (*IsCSP*). At level 1, each node is set with a single frequent item; at level *k*, each node is set with a *k*-pattern sequence. Recursively, there are nodes at the next level (*k+1*) after a *k*-pattern sequence is extended with a single frequent item. There are two ways to extend a *k*-pattern sequence, namely sequence extension and itemset extension [3]. In sequence extension, a single frequent item from *I* is added to the *k*-pattern sequence as a new itemset, increasing the size of the sequence. A sequence α is a prefix of all sequence-extended sequences of α, and α is the prefix of all subnodes of the nodes that are sequence-extended in α. In itemset extension, single frequent item from *I* that is greater than all items in the last itemset is added to the last itemset in the *k*-pattern sequence. The size of itemset-extended sequences does not change and *α* is an incomplete prefix of all subnodes of itemset-extended nodes in *α*.

4. PROPOSED ALGORITHM FOR MINING CLOSED SEQUENTIAL PATTERNS

In this section, the extension of a sequence on the prefix tree by performing a depth-first search and the attribute of closed seuquential patterns are used to produce an algorithm for generating all closed sequential patterns. Using the prefix tree, new sequences, which are children nodes *Cnode*, can be easily created by appending an item to the last position of a parent node *Pnode* as an itemset extension or a sequence extension. When a new node *Pnode* is created, if *Sup(Pnode) = Sup(Cnode)*, then we set the *IsCSP* of *Cnode* to false and the *IsCSP* of *Pnode* to true.

First, the algorithm initializes the prefix tree *pretree* with the root node being null and children nodes being sequential 1-patterns with its *IsCSP* field as *true*. Each child node *cn* on *pretree* is considered as a root node for *EXTEND_TREE(cn, pretree)* function to create its children nodes and extend the *pretree* tree. In this function, each child node of root node *P* is created by itemset extension or sequence extension. To represent candidate sequences as well as determine the frequency for each candidate, it uses the prime block encoding approach and the join operations over the prime blocks in [3]. With each new child node is created *Pnew* from *P*, if *Sup(Pnew)=Sup(P)*, then the value of *Pnew.IsCSP* is set to *true* and the value of *P.IsCSP* is set to *false*. The *CHECK_CLOSED(Pnew, pretree)* function is called to update closed sequential patterns on the *pretree* tree. Finally, the algorithm returns the corresponding *pretree* tree with the sequential patters which have the corresponding IsCSP values. The details of the proposed algorithm are introduced in Figure 1.

```
Input: SD, minSup.
Output: Prefix tree that contains Sequential/Closed sequential patterns.
Method:
MCSP-PreTree(SD, minSup)
    pretree ← ∅;
    SPs ← all frequent 1-pattern sequences;
    for each pattern P in SPs
            Add P into pretree as a child node;
    for each child node r in pretree
            EXTEND_TREE(r, pretree);//extend tree with root node r
    return pretree;
//.....................................
EXTEND_TREE(Root, pretree)
    EXTEND_ITEMSET(Root, pretree);//create a new child node of root by itemset extension
    EXTEND_SEQUENCE(Root, pretree );//create a new child node of root by sequence extension
    For each node $P_i$ that is an itemset extension of Root
            EXTEND_TREE($P_i$, pretree);// recursively call to extend tree with root node $P_i$
    For each node $P_s$ that is a sequence extension of Root
            EXTEND_TREE($P_s$, pretree); // recursively call to extend tree with root node $P_s$
 // ....................................................
EXTEND_ITEMSET(P, pretree)
```

```
            For each pattern $P_i$ in SPs with $P_i$>last item in last itemset of P
                Pnew is a new node created by adding $P_i$ into last position in last itemset of P and using block encoding
              based on prism factorization in [3] to count the support;
              If (sup(Pnew) ≥ minSup)
                    Set Pnew as a closed pattern;
                  If (sup(P) = sup(Pnew))
                        Set P as a non-closed pattern;
                  Else
                            CHECK_CLOSED(Pnew, pretree);//check whether there exist subsequences of
                        Pnew in pretree which are closed sequential patterns.
                  Add Pnew into pretree as a itemset-extended child node of P;
    //..................................................


    EXTEND_SEQUENCE(P, pretree)
      For each 1-pattern $P_i$ in SPs
                Pnew is a new node created by adding $P_i$ as last itemset into P and using block encoding based on
              prism factorization in [3] to count the support;
                If (sup(Pnew) ≥ minSup)
                    Set Pnew as a closed pattern;
                  If (sup(P) = sup(Pnew))
                        Set P as non-closed pattern;
                  Else
                        CHECK_CLOSED(Pnew, pretree);//check whether there exist subsequences of Pnew in
              pretree which are closed sequential patterns
                  Add Pnew into pretree as a sequence-extended child node of P;
    //..................................................
CHECK_CLOSED (Pnew, pretree)
            For each node P in pretree
                If Pnew is a supersequence of P
                        If P is a closed pattern and sup(P) = sup(Pnew)
                                Set P as a non-closed pattern;
                subtree = the tree which rooted at P;
                CHECK_CLOSED(Pnew, subtree);
```

*Figure 1***.** Algorithm for generating set of closed sequential patterns.

5. EXPERIMENTAL RESULTS

Figures 2–4 illustrate the process of mining closed sequential patterns on prefix tree using the proposed algorithm for the sequence database as presented in Table 1, where *minSup* = 50 %. First, the root node of prefix tree is null ({}) with each child node containing: sequential pattern, support value, and *IsCSP* field that contains one of two values (true T or false F). Level 1 of the prefix tree contains nodes with sequential 1-pattern, their support values and the values of *IsCSP* set to *true* as in Figure 2, for example node 〈(B)〉:5:T . Consider node 〈(A)〉:4:T in Figure 2, the children nodes of 〈(A)〉:4:T are created by itemset extension or sequence extension, which include sequential pattern nodes: 〈(A)(B)〉:3:T, 〈(A)(C)〉:3:T, 〈(AB)〉:4:T. The support value of the child node containing sequential pattern ⟨(AB)⟩ is 4, which is same as the support value of the parent node containing ⟨(A)⟩, according to the definition of closed sequential pattern, then ⟨(A)⟩ is not a closed sequential pattern, so the *IsCSP* value of ⟨(A)⟩ is set to *false* T. After updating nodes in the prefix tree, we have the prefix tree as in Figure 3. The above process is repeated for all children nodes of ⟨(A)⟩ and all remaining nodes on the prefix tree. The final experimental results are shown in Figure 4. The experimental results in Figure 4 show that the number of closed sequential patterns is always smaller than the number of sequential patterns, where the complete set of closed sequential patterns consists of only 8 sequences while the whole set of sequential patterns consists of 21 sequences.

Table 1. Sequence database.

SID	Sequence
1	⟨(AB)(B)(B)(AB)(B)(AC)⟩
2	⟨(AB)(BC)(BC)⟩
3	⟨(B)(AB)⟩
4	⟨(B)(B)(BC)⟩
5	⟨(AB)(AB)(AB)(A)(BC)⟩

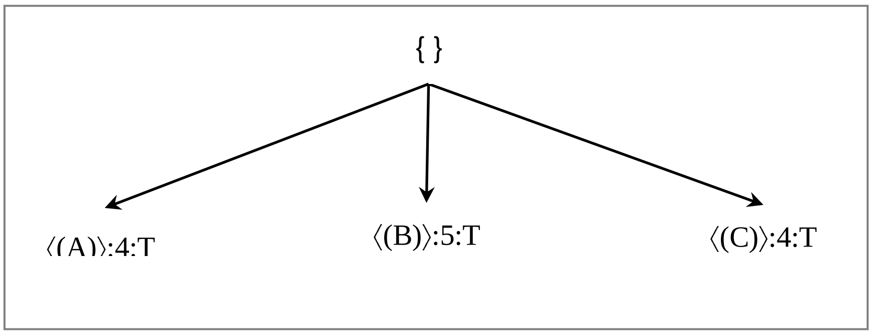

Figure 2. Level 1 of the prefix tree (each node contains: sequential pattern, support, and IsCSP).

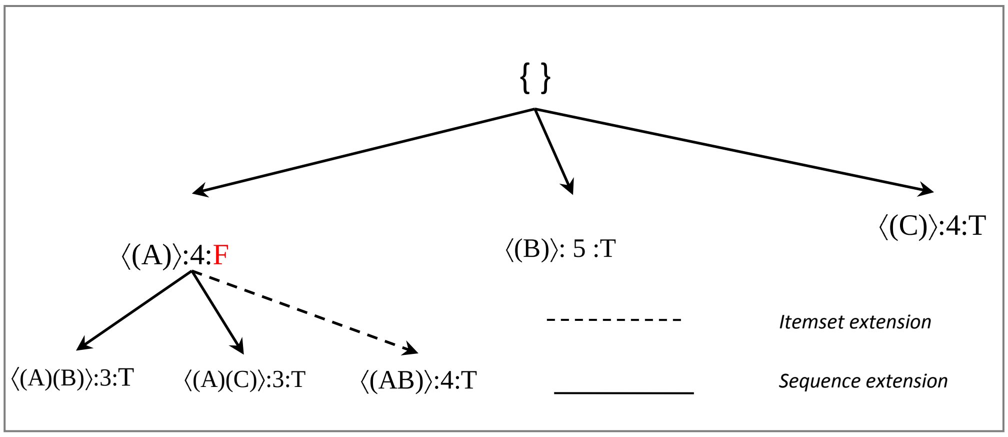

Figure 3. Updating nodes in the prefix tree after the children nodes of ⟨(A)⟩ node are created.

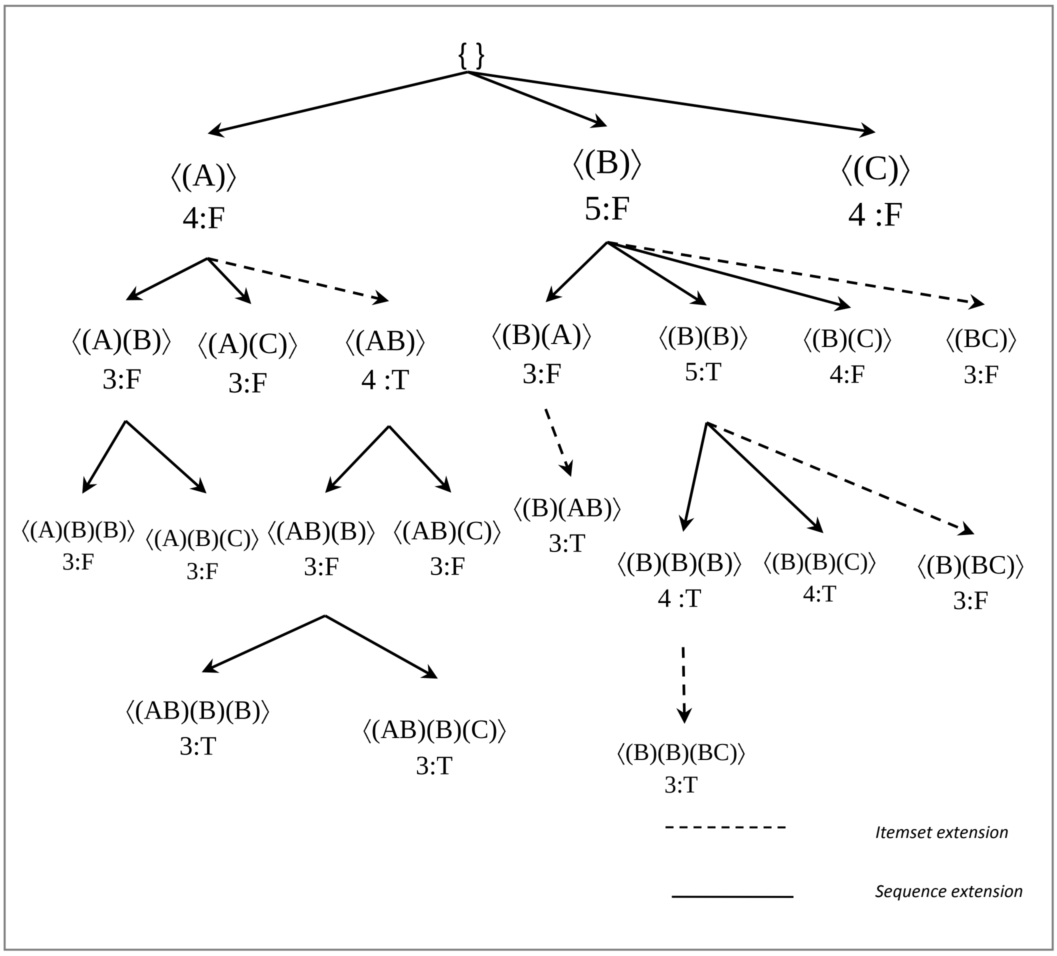

Figure 4. The prefix tree of sequence database in Table 1 with *minSup* = 50 %, which has the corresponding *IsCSP* values.

5. CONCLUSIONS AND FUTURE WORK

This paper proposed an algorithm for mining closed sequential patterns. This algorithm combined the parent–child relationship between nodes on prefix tree and the definition of closed sequential pattern to determine whether a sequential pattern is a closed sequential pattern by adding *IsCSP* field into each sequential pattern node on the prefix tree. The algorithm also applied the prime block encoding approach and the join operations over the prime blocks in [3] for generating candidate sequences and determining the frequency for each candidate. Experimental results are examined on the small sequence database also showed that the number of closed sequential patterns is much smaller than that of the sequential patterns.

In future, based on this algorithm, we will perform more experimental results with the larger sequence databases such as synthetic and real databases, namely C6T5S4I4N1kD1k, Chess, and Mushroom and so on. C6T5S4I4N1kD1k was generated using the synthetic data generator developed by IBM to mimic transactions in a retail environment Chess and Mushroom databases were downloaded from http://fimi.ua.ac.be/data/. Besides, the lattice-base approach has been proposed for mining association rules and classification association rules in recent years [5, 13]. We will study how to apply this approach for mining sequential patterns and sequential rules in the future.

Acknowledgements. This research is funded by Viet Nam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2015.07.

REFERENCES

1. Agrawal R. and Srikant R. - Mining Sequential Patterns. In: Proc. of 11th Int'l Conf. Data Engineering, DC, USA, (1995) 3 –14.
2. Ayres J., Gehrke J.E., Yiu T. and Flannick J. - Sequential Pattern Mining using a Bitmap Representaion. In: SIGKDD Conf., NY, USA, (2002) 1–7.
3. Gouda K., Hassaan M. and Zaki M.J. - PRISM: A Primal-Encoding Approach for Frequent Sequence Mining. Journal of Computer and System Sciences **76**(1) (2010) 88-102.
4. Huang G.-Y., Yang F., Hu C.-Z. and Ren J.-D. - Fast Discovery of Frequent Closed Sequential Patterns based on Positional. Proc. of the International Conference on Machine Learning and Cybernetics, ICMLC 2010, Qingdao, China, (2010) 444 – 449.
5. Nguyen L.T.T., Vo B., Hong T.P. and Thanh H.C. - Classification based on association rules: A lattice-based. Expert Systems with Applications, 39(13), (2012) 1135 –1136.
6. Pei J., Han J. and Mao R. - CLOSET: An efficient algorithm for mining frequent closed itemsets. In DMKD'01 workshop, Dallas, TX, (2001) 21 – 30.
7. Pham T.-T., Luo J. and Vo B. - An effective algorithm for mining closed sequential patterns and their minimal generators based on prefix trees. International Journal of Intelligent Information and Database Systems **7**(4) (2013) 324-339.
8. Pham T.T., Luo J., Hong T.-P. and Vo B. - An Efficient Method for Mining Non-Redundant Sequential Rules Using Prefix-Trees, Engineering Applications of Artificial Intelligence **32** (2014), 88 - 99.
9. Pei J., et al - Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. In: IEEE Trans. Knowledge and Data Engineering **16**(10) (2004) 1424 –1440.
10. Srikant R. and Agrawal R. - Mining Sequential Patterns: Generalizations and Performance Improvements. In: Proc. of 5th Int'l Conf. Extending Database Technology, London, UK, (1996) 3–17.
11. Thilagu M., Nadarajan R., Ahmed M.S.I. and Bama S.S. - PBFMCSP: Prefix Based Fast Mining of Closed Sequential Patterns. The International Conference on Advances in Computing, Control, and Telecommunication Technologies ATC'09, Trivandrum, Kerala, India,(2009) 484 – 488.
12. Tzvetkov P., Yan X. and Han J. - TSP: Mining Top -K Closed Sequential Patterns. Knowl. Inf. Syst., (2005) 438-457.
13. Vo B. and Le B. - Interestingness measures for association rules: Combination between lattice and hash tables. Expert Systems with Applications **38**(9) (2011) 1630 - 1640.
14. Wang J. and Han J. - BIDE: Efficient mining of frequent closed sequences. In proc of the 20th Int' Conf on Data Engineering (ICDE95): IEEE Computer Society Press, DC, USA, (2004) 79-91.

15. Yan X., Han J. and Afshar R. - CloSpan: Mining closed sequential patterns in large datasets. Proc of the 3th SIAM International Conference on Data Mining, San Francisco, CA, USA: SIAM Press, (2003) 166 -177.
16. Zaki M.J. - SPADE: An Efficient Algorithm for Mining Frequent Sequences. In: Machine Learning Journal **42**(1/2) (2000) 31- 60.
17. Zaki M.J. and Hsiao C. - CHARM: An efficient algorithm for closed itemset mining, In SDM '02, Arlington, VA, (2002) 457 - 473.