# A NOVEL ALGORITHM FOR MINING CLOSED SEQUENTIAL PATTERNS

V. Purushothama Raju[1] and G.P. Saradhi Varma[2]

[1]Research Scholar, Dept. of CSE,
Acharya Nagarjuna University, Guntur, A.P., India
[2]Department of Information Technology
S.R.K.R. Engineering College, Bhimavaram, A.P., India

***ABSTRACT***

*Sequential pattern mining algorithms produce an exponential number of sequential patterns when mining long patterns or at low support thresholds. Most of the existing algorithms mine the full set of sequential patterns. However, it is sufficient to mine closed sequential patterns from which the total set of sequential patterns can be derived and the closed sequential patterns set is more compact than the sequential patterns set. In this paper, we propose a novel algorithm NCSP for mining closed sequential patterns in large sequences databases. To the best of our knowledge, our algorithm is the first algorithm that utilizes vertical bitmap representation for closed sequential pattern mining. The results show that the proposed algorithm NCSP can find closed sequential patterns efficiently and outperforms CloSpan by an order of magnitude.*

***KEYWORDS***

*Data mining, sequential pattern mining, closed sequential pattern mining, sequence database*

## 1. INTRODUCTION

Data mining finds unknown and useful knowledge from large databases or data warehouses. Many techniques were developed to discover the knowledge. One of the important techniques is sequential pattern mining. The sequential pattern mining technique finds the full set of sequential patterns in a sequence database.

Agrawal and Srikant [1] developed the sequential pattern mining. Later a large number of works concentrated on it, because of its applicability to a huge number of applications including market analysis, web log analysis, DNA sequences and network intrusion detection. The problem of sequential pattern mining is to find all sequential patterns whose support is greater than minimum support for a given a sequence database and the minimum support threshold.

Most of the sequential pattern mining methods are based on the Apriori property [2] i.e. all subpatterns of a frequent pattern must be frequent. It leads to the problem of producing an exponential number of sequences, which is not acceptable when the database contains long sequences. For example, a frequent long sequence $\{(a_1)(a_2)....(a_{50})\}$ will generate $2^{50}$ - 1 frequent subsequences which are basically redundant patterns.

The same problem also occurs in mining frequent itemsets. Over the years, many algorithms were developed to find frequent itemsets. However, most of the algorithms neglect the output quality

DOI : 10.5121/ijdkp.2015.5104 

and generate a large number of patterns, many of the generated patterns are redundant. For example, a long frequent itemset with size n can generate 2n candidate subsets. However, the actual occurrence of all itemsets can be determined from the closed itemsets. Closed itemsets are compact than the frequent itemsets and also include entire information. Therefore it is sufficient to find only closed itemsets.

Previous studies concluded that a sequential pattern mining algorithm need not mine complete frequent sequences but only the closed sequences for reducing the memory and runtime. Because the closed sequential pattern mining produces more compact output without losing any information and provides better efficiency. It can also be used to obtain the total set of sequential patterns.

Unfortunately, most of the methods proposed for closed itemset mining are not directly applicable for closed sequential pattern mining. Because subsequence testing in closed sequential pattern mining needs order matching which is more complex than subset testing in closed itemset mining and the search space of closed sequences is much higher than closed itemsets. Even though, closed itemset mining was studied greatly, only a few algorithms have been developed for closed sequential pattern mining.

In this paper, we propose a novel algorithm NCSP for mining closed sequential patterns. Our algorithm is the first algorithm that uses vertical bitmap representation [3] for closed sequential pattern mining. The experimental results indicate that the proposed algorithm NCSP can find closed sequential patterns efficiently and runs faster than CloSpan[4].

The contributions of this paper are as follows:

1. We propose a novel algorithm that utilizes a vertical bitmap representation for efficient counting of support.
2. We show the performance of our algorithm on several real and synthetic datasets.

The rest of this paper is organized as follows. In section 2, we discuss the related work. In section 3, we present the problem definition. In section 4, we present the proposed method including vertical bitmap representation, lexicographic sequence tree and algorithm. In section 5, we report the performance evaluation of our proposed algorithm. Finally, we conclude the work in section 6.

## 2. RELATED WORK

Closed sequential pattern mining is associated with sequential pattern mining and closed itemset mining. Sequential pattern mining is used to discover the total set of sequential patterns in a sequential database. Sequential pattern mining was first proposed by R. Agrawal and R. Srikanth in [1]. The same authors also proposed a generalized algorithm for sequential pattern mining GSP [5] to reduce the search space for finding frequent sequences. Since then, many algorithms have been developed for sequential pattern mining.

Algorithms such as GSP, SPAM [3] and SPADE [6] were developed based on Apriori method. Apriori-based methods use candidate generate-and-test framework, which make use of the downward closure property. Algorithms like FreeSpan [7] and PrefixSpan [8] are based on pattern-growth model. Pattern-growth type algorithms follow an incremental mechanism for producing possible frequent sequences and construct projections of the database for decreasing the search space.

SPADE, PrefixSpan and SPAM algorithms increase the efficiency of sequential pattern mining in terms of time and space complexity. SPADE implements breadth-first search where as PrefixSpan and SPAM implement depth-first search. SPADE employs vertical data format and produces the sequential patterns using a simple join on id-lists. PrefixSpan uses horizontal data format and produces the sequential patterns using the pattern growth model. SPAM uses vertical bitmap representation and runs faster than PrefixSpan and SPADE. But, SPAM consumes more memory space than the above two methods.

Closed itemset mining was developed for generating closed itemsets that does not have any supersets with the same support. Closed itemset mining generates smaller result set comparing to frequent itemset mining. Close [9] mines closed itemsets and it is based on Apriori approach. Close first employs bottom-up search to find out the generators and then calculates the closure of all the generators.

FPclose[10] mines closed itemsets and it is based on FP-growth method. It uses an FP-tree during the mining process to store the frequency count of the entire dataset. To test closure of a frequent itemset, it uses another data structure Closed Frequent Itemset tree for tracking all closed itemsets, and to decrease the search space and the no of subset testing actions.

CLOSET[11] and CHARM [12] implement space efficient depth first search. CLOSET uses FP-tree to create compressed database representation for mining closed itemsets. CHARM mines closed itemsets using a compact vertical tid list structure known as diffset.

CLOSET+[13] uses item skipping and subset-checking techniques for mining closed itemsets. Item skipping technique prunes the search space to speed up the mining. The subset-checking technique reduces the memory usage and performs the closure-checking. CLOSET+ outperforms CLOSET and CHARM in terms of execution time, memory utilization and scalability.

Another recent focus in sequential pattern mining is to mine the closed sequential patterns instead of mining the complete set of sequential patterns for achieving a more compact output set with better efficiency. A pattern is called as a closed pattern if it does not have any super pattern with the same support.

The two well known algorithms in closed sequential pattern mining are CloSpan [4] and BIDE [14]. Both CloSpan and BIDE adopt the framework of PrefixSpan. Similar to most of the closed itemset mining algorithms, CloSpan uses a candidate maintenance-and-test paradigm. CloSpan performs the mining in two stages. In the first stage it produces a closed sequential pattern candidate set and keeps it in a prefix sequence lattice. In the second stage, it performs post pruning to eliminate non closed sequential patterns. CloSpan uses an efficient search space pruning method, known as equivalence of projected databases. It performs subsequence/super sequence checking efficiently by using the size of the projected databases as the hash key.

CloSpan generates less number of patterns than the sequential pattern mining methods while preserving the same expressive power and runs faster than PrefixSpan. CloSpan works under candidate maintenance-and-test paradigm, hence it is not scalable because a large number of closed sequential pattern candidates will occupy more memory space and lead to huge search space for checking the closure of new patterns, particularly for low support threshold values or long patterns.

BIDE produces closed sequential patterns without candidate maintenance. It uses BI-Directional Extension closure checking scheme for eliminating nonclosed sequences. It prunes the search space by using the BackScan pruning method. BIDE consumes less memory and runs faster than

the previously developed closed sequential pattern mining algorithms, particularly when the support is less. It provides better scalability. BIDE is a computational consuming approach because it requires more no of database scans for the bi-directional closure checking and the BackScan pruning.

FMCSP [15] adopts a breadth-first method and it can output the frequent closed patterns online. It eliminates the disadvantage of the candidate maintenance-and-test paradigm and produces more compact search space comparing to the previously introduced closed pattern mining algorithms. FMCSP uses equivalence class to reduce the runtime.

## 3. PROBLEM DEFINITION

In this section, we first introduce some basic concepts and then define the problem of closed sequential pattern mining.

Let $I = \{i_1,i_2,\ldots,i_m\}$ be a set of all items. A subset of $I$ is called an itemset. A sequence $S = (k_1,k_2,\ldots, k_n)$ ($k_i \subseteq I$) is an ordered list of itemsets. The items in each itemset are sorted in alphabetic order. The length of the sequence is the total number of items in the sequence. A sequence $S_1 =(a_1,a_2,\ldots..,a_m)$ is a subsequence of another sequence $S_2 =(b_1,b_2,\ldots.,b_n)$, denoted as $S_1 \sqsubseteq S_2$, if there exists integers $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ and $a_1 \subseteq b_{i1}$ , $a_2 \subseteq b_{i2}$ , . . . , and $a_m \subseteq b_{im.}$ We call $S_2$ as a super-sequence of $S_1$ and $S_2$ contains $S_1$.

A sequence database, $SD=\{S_1,S_2,\ldots,S_n\}$, is a set of sequences and each sequence has an id. The size of the sequence database SD is the total number of sequences in the SD. The support of a sequence $\alpha$ in a sequence database SD is the no of sequences in SD which contain $\alpha$.

Given a minimum support threshold m_sup, a sequence $\alpha$ is a sequential pattern on SD if support of $\alpha$ is greater than m_sup. We call a sequence $\alpha$ as a closed sequential pattern if $\alpha$ is a sequential pattern and there exists no proper super sequence of $\alpha$ with the same support. The problem of closed sequential pattern mining is to find the complete set of closed sequential patterns above a minimum support threshold m_sup for an input sequence database SD.

Table 1. A sample sequence database

| Sid | Sequence |
|---|---|
| 1 | (ab)(cde) |
| 2 | (bc)(cd)(e) |
| 3 | (bd)(ce) |

Table 1 represents a sample sequence database. In each itemset, the items are arranged in alphabetic order. If m_sup=2 , the closed sequential pattern set contains 6 sequences { **(d):3, (b)(c):3, (b)(e):3, (d)(e):2, (b)(cd):2, (b)(ce):2**} and the corresponding sequential pattern set contains 12 sequences{**(b):3, (c):3, (d):3, (e):3, (b)(e):3, (b)(c):3, (b)(d):2, (d)(e):2, (ce):2, (cd):2, (b)(ce):2, (b)(cd):2**}. It shows that closed sequential pattern set contains less number of patterns than sequential pattern set.

Given a sequence $S=(k_1,\ldots,k_m)$and an item $\alpha$, $S\Delta\alpha$ means S concatenates with $\alpha$. It can be itemset extension $S\Delta_i\alpha=(k_1,\ldots,k_m \cup (\alpha))$ or sequence extension $S\Delta_s\alpha=(k_1,\ldots,k_m,(\alpha))$. For example (bc) is an itemset extension of (b) and (c)(d) is a sequence extension of (c).

# 4. PROPOSED METHOD

In this section we discuss the vertical bitmap representation for storing the sequences in bitmap form, lexicographic sequence tree and the proposed algorithm.

## 4.1 Vertical Bitmap Representation

To increase the efficiency of support counting, we adopt a vertical bitmap representation for the dataset. A vertical bitmap is produced for every item in the database, and each itemset in the sequence is assigned with a bit in every bitmap. If item x appears in itemset k, then the corresponding bit in the bitmap is set to 1. Otherwise, the bit is assigned with the value 0.

We partition the bitmaps according to the itemsets in each sequence as shown in Fig. 1. The first partition contains 2 itemsets of sequence1, the second partition contains 3 itemsets of sequence2 and the third partition contains 2 itemsets of sequence3. If itemset m is before itemset n in a sequence, then the index of the bit m is made smaller than the index of the bit n. The bitmap for the itemset {m, n} is obtained by performing the bitwise AND operation between bitmaps for item m and n. Support of a sequence is obtained by checking whether the corresponding bitmap partition includes all zeros or not.

Based on the lengths of sequences, we partition the sequences into individual sets. If the length of a sequence is between $2^n + 1$ and $2^{n+1}$ then it is considered as a $2^{n+1}$ bit sequence. The minimum value of $n$ is 1. Each set of $2^n$ bit sequences is considered as a separate bitmap, and in that bitmap the length of each section is $2^n$ bits.

Fig. 1 gives the bitmap representation of the sample sequence database shown in Table 1. Each partition in the vertical bitmap corresponds to a customer's sequence. The itemset 1 of sequence 1 contains items a and b, hence the bits in the bitmaps of a and b of the corresponding itemset are set to one and the bits for remaining items c, d and e are set to zero in the bitmaps of c,d and e.

| Sid | Ino | (a) | (b) | (c) | (d) | (e) |
|---|---|---|---|---|---|---|
| **1** | **1** | 1 | 1 | 0 | 0 | 0 |
| **1** | **2** | 0 | 0 | 1 | 1 | 1 |
| **2** | **1** | 0 | 1 | 1 | 0 | 0 |
| **2** | **2** | 0 | 0 | 1 | 1 | 0 |
| **2** | **3** | 0 | 0 | 0 | 0 | 1 |
| **3** | **1** | 0 | 1 | 0 | 1 | 0 |
| **3** | **2** | 0 | 0 | 1 | 0 | 1 |

Figure 1. Vertical bitmap representation of a sample sequence database

## 4.2 Lexicographic Sequence Tree

We use lexicographic sequence tree to explain our algorithm. The elements in the tree are arranged in lexicographic order i.e. in alphabetical order. If item $a$ is before item $b$ in the order, then we denote it as $a \leq b$. Assume all sequences in the database are organized in the lexicographic sequence tree. The root of the tree is denoted as $\phi$. If x is a node in the tree and its children are all nodes x′ then the lexicographic ordering between x and x′ is represented as $x \leq x'$.

New sequences are generated by using sequence-extension and itemset-extension approaches. The sequence-extension approach produces a new sequence by adding a single item to the end of the existing sequence. The itemset-extension approach produces a new sequence by adding a single item to the last itemset in the existing sequence. For example, {(ab),(de),(f)}is a sequence-extended sequence of {(ab),(de)} and {(ab),(def)} is an itemset-extended sequence of {(ab),(de)}. Each node p in the tree is coupled with two sets: $S_p$, and $I_p$. $S_p$ includes sequence-extended sequence candidates of node p and $I_k$ includes itemset-extended sequence candidates of node p.

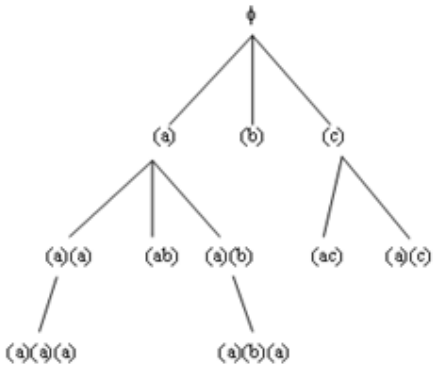


Figure 2. A sample lexicographic sequence tree.

Fig. 2 shows a sample lexicographic sequence tree. The root has a null sequence and each lower level n of the tree contains sequences of length n. Each child node in the tree is obtained by using sequence-extension and itemset-extension approaches.

Our algorithm NCSP uses depth-first search (DFS) approach for traversing the lexicographic sequence tree. At each node the support of each sequence-extended sequence and each itemset-extended sequence is calculated. Sequences whose support is greater than or equal to minimum support are gathered and DFS is repeated on them.

## 4.3 Algorithm Development

In this subsection, we explain our proposed NCSP algorithm. NCSP has two phases. The first phase generates Closed Sequence Candidates (CSC) and the second phase performs post-pruning to eliminate all non-closed sequences from CSC to finally generate closed sequential patterns. Algorithm 1 shows the pseudo code corresponding to these two phases.

**Algorithm1: N**CSP

Input: A sequence database SD and minimum support min_sup.
Output: The complete set of closed sequential patterns.
1. Remove infrequent items in SD
2. Construct vertical bitmap for each item in the database
3. S1 = frequent 1-sequnces
4. CSC = ϕ
5. for each i in S1 do
6. $S_j$ = frequent 1-sequences greater than i
7. $CSC_i$ = Generate_candidates (i, S1, $S_j$)
8. CSC = CSC ∪ $CSC_i$
9. end for
10. Eliminate nonclosed sequences in CSC using hashing technique.

**Algorithm 2:** Generate_candidates(S, $S_n$, $I_n$)
Input: A sequence S, sequence extension set $S_n$ and itemset extension set $I_n$.
Output: Closed sequence candidates.
1. Perform pruning by doing sub pattern and super pattern checking of S.
2. for each i in $S_n$ do
3. Generate closed sequence candidates by performing sequence extensions
4. end for
5. Call Generate_candidates for each sequence extended candidate
6. for each i in $I_n$ do
7. Generate closed sequence candidates by performing itemset extensions
8. end for
9. Call Generate_candidates for each itemset extended candidate
10. Return closed sequence candidates

Our proposed algorithm NCSP first scans the database to remove infrequent items in the database. Next, for each item in the database it builds a vertical bitmap and each bitmap includes a bit for every element in the sequence. Then it finds all frequent 1-sequences, after that for each frequent 1-sequence the method Generate_candidates is called to generate the candidates. CSC is obtained when this process is done for all of the frequent 1-sequences and finally it performs closure checking using hashing technique to eliminate nonclosed sequential patterns in CSC.

Algorithm 2, Generate_candidates, executes recursively for generating the candidates by means of sequence extensions and itemset extensions, and returning a part of CSC relative to the pattern S taken as parameter. The method takes two sets ($S_n$ and $I_n$) to do sequence extensions and itemset extensions respectively. To prune the search space, the algorithm first checks if the current pattern S can be discarded or not.

The pruning is implemented by two methods: 1) sub-pattern checking and 2) super-pattern checking. The first method occurs when we find a pattern which is a subsequence of a pattern previously found with the same support value. In that case, we can avoid exploring this new branch in the tree for this new pattern. The second method occurs when we find a pattern that is a super-sequence of another pattern previously found with the same support value. In this case we transplant the descendants of the previous pattern to the node of this new pattern.

To eliminate nonclosed sequential patterns we use a hash function with the support of a pattern as key and the pattern itself as value. If two patterns have the same support we check if one contains the other, and if this condition is satisfied, we remove the shorter pattern.

## 5. PERFORMANCE EVALUATION

To evaluate the effectiveness of our proposed NCSP algorithm, we performed an extensive performance study on both real and synthetic data sets with various kinds of sizes and data distributions. All experiments were performed on a 2GHz Intel Core2 Duo processor PC with 1GB main memory running Microsoft Windows XP. The algorithms NCSP and CloSpan were implemented in Java and were executed using different support values.

In our experiments we used a real world dataset MSNBC and two synthetic datasets. MSNBC is a click stream data taken from the UCI repository. It contains 9,89,818 sequences. The shortest sequences were eliminated to maintain only 31,790 sequences. The total individual items in this dataset are 17. The average no of itemsets per sequence is 13.33. The average no of different items per sequence is 5.33. The characteristics of the MSNBC dataset are shown in Table 2. We

generated the synthetic datasets using SPMF[16] framework. The characteristics of the two synthetic datasets are given in Table 3.

Table 2. Characteristics of the MSNBC dataset

| S. No. | Characteristic | Value |
|---|---|---|
| 1 | No of sequences | 31790 |
| 2 | No of distinct items | 17 |
| 3 | Average number of itemsets per sequence | 13.33 |

Table 3. Characteristics of the synthetic datasets

| S. No. | Characteristic | Dataset1 Value | Dataset2 Value |
|---|---|---|---|
| 1 | No of sequences | 20000 | 15000 |
| 2 | No of distinct items | 20 | 30 |
| 3 | No of items per itemset | 4 | 3 |
| 4 | No of itemsets per sequence | 6 | 8 |

To evaluate the performance of the algorithm, three sets of experiments were conducted. The first set compares the runtime performance of NCSP with CloSpan using real world dataset MSNBC for different support values. The second and third sets compare the runtime performance of NCSP with CloSpan using synthetic datasets for different support values.

Fig. 3 shows the results of runtime performance using the real world dataset MSNBC. The x-axis is the minimum support, while the y-axis is the algorithms runtime. The support values are set from 0.1 to 0.6. Because of the usage of vertical bitmap representation technique our proposed algorithm NCSP runs faster than CloSpan. The runtime time is high at low support threshold due to the generation of more number of patterns. The runtime decreases when the support threshold increases due to the decrease of patterns.

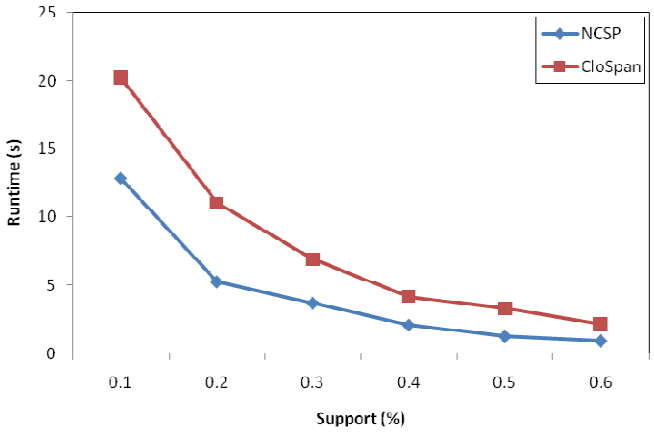


Figure 3. Performance comparison using MSNBC dataset.

Fig. 4 and Fig. 5 show the results of runtime performance using the synthetic datasets. The x-axis is the minimum support, while the y-axis is the algorithms runtime. The support values are set from 0.1 to 0.6. Because of the usage of vertical bitmap representation technique our proposed algorithm NCSP outperforms CloSpan.

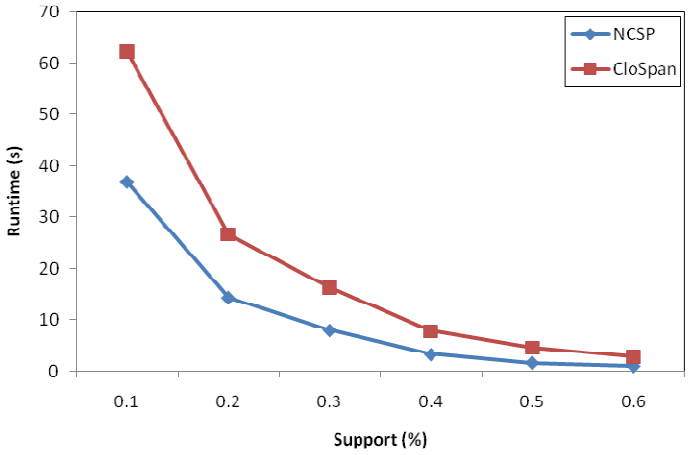


Figure 4. Performance comparison using synthetic dataset1

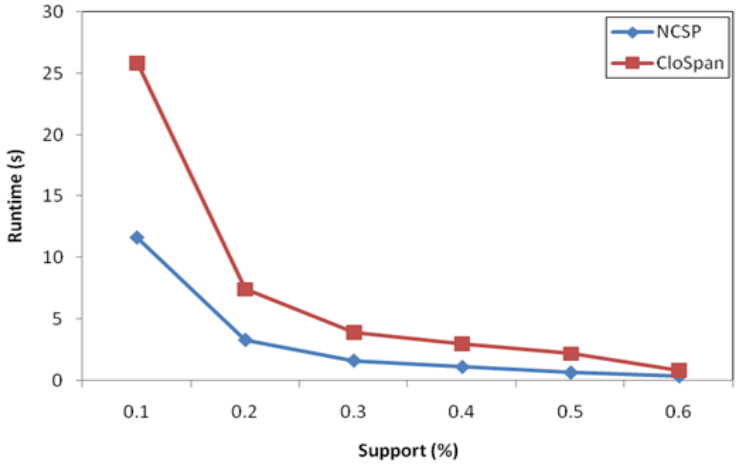


Figure 5. Performance comparison using synthetic dataset2

## 6. CONCLUSION

In the past, the ways to uncover sequential patterns from sequence data resulted in a large number of research efforts. Sequential pattern mining algorithms generate huge number of sequential patterns and most of them are redundant. To solve this problem, the closed sequential pattern mining is proposed.

In this paper, we propose a novel algorithm NCSP for mining closed sequential patterns from sequence data. To increase the speed of support counting NCSP uses vertical bitmap representation.

The experimental results show that NCSP outperforms CloSpan on both real and synthetic datasets. The pruning strategy and closure checking strategy are more effective in reducing the search space and eliminating nonclosed sequential patterns. Further research problems include how to incorporate constraints for closed sequential pattern mining[17] and mining multidimensional closed sequential patterns[18].

## REFERENCES


[1] R. Agrawal and R. Srikant, "Mining sequential patterns," Proceedings of ICDE '95, pp. 3- 14, Mar. 1995.

[2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules in large databases, " Proc. 20th Int'l. Conf. Very Large Data Bases (VLDB 94), pp. 487-499, Sept. 1994.

[3] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential pattern mining using a bitmap representation," Proceedings of ACM SIGKDD '02, pp. 429-435, July 2002.

[4] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining closed sequential patterns in large databases," Proceedings of SIAM's SDM '03, pp. 166-177, May 2003.

[5] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," Proceedings of EDBT '96, pp. 3-17, Mar. 1996.

[6] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," Kluwer Academic Publisher's Machine Learning, vol. 42, pp. 31-60, 2001.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, "FreeSpan: Frequent pattern-projected sequential pattern mining," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '00), pp. 355-359, Aug. 2000.

[8] J. Pei, J. Han, B. Mortazavi-Asl, and et al., "PrefixSpan : Mining sequential patterns efficiently by prefix-projected pattern growth," Proc. Int'l Conf. Data Engineering (ICDE '01), pp. 215-224, Apr. 2001.

[9] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules," 7th Int'l Conf. on Database Theory, pp. 398-416, Jan. 1999.

[10] Grahne G and Zhu J F, "Fast algorithm for frequent itemset mining using FP-Trees," IEEE Trans. on Knowledge and Data Engineering, vol. 17, no. 10, pp. 1347-1362, 2005.

[11] J. Pei, J. Han, and R. Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets," Proceedings of ACM DMKD '00, pp. 21-30, May 2000.

[12] M. Zaki and C. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," Proceedings of SIAM's SDM '02, pp. 457-473, Apr. 2002.

[13] J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the best strategies for mining frequent closed itemsets," Proceedings of ACM SIGKDD '03, pp. 236-245, Aug. 2003.

[14] J. Wang, J. Han, and Chun Li, "Frequent closed sequence mining without candidate maintenance," IEEE TKDE, vol. 19, no. 8, pp. 1042-1056, Aug. 2007.

[15] Nancy P. Lin, Wei-Hua Hao, Hung-Jen Chen, Hao-En Chueh and Chung-I Chang, "Fast mining of closed sequential patterns," WSEAS Transactions on Computers, vol. 7, no. 3, Mar. 2008.

[16] Fournier-Viger P., An Open-Source Data Mining Library, http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php, 2008, Accessed 20 July 2014.

[17] Takei H. and Yamana H., "IC-BIDE: Intensity Constraint-Based Closed Sequential Pattern Mining for Coding Pattern Extraction," IEEE 27th International Conference on Advanced Information Networking and Applications (AINA 2013), pp. 976-983, Mar. 2013.

[18] Panida Songram and Veera Boonjing, "Closed multidimensional sequential pattern mining," Int. J. Knowledge Management Studies, vol. 2, no. 4, pp. 460-479, 2008.