Modified Sequential Pattern Mining Using Direct Bit Position Method

Dr. K. Subramanian, M.Sc., M.Phil, Ph.D[1], E. Elakkiya[2]

[1]Assistant Professor, VSS Government Arts College, Pulankurichi – 630405, Tamil Nadu, India

[2]Research Scholar, PG & Research Department of computer Science, J.J College of Arts and Science, Pudukkottai -622004, Tamil Nadu, India

Abstract: ***A new algorithm for mining sequential patterns was proposed in this paper. The proposed algorithm employs efficient direct bit position manipulations techniques for mining sequential data with long sequences. The prominent feature of the proposed algorithm DBP-SPAM is that the frequent item sets are discovered by bit position pruning technique and the experimental evaluation showcased that the proposed algorithm outperforms the previous algorithms up to an order of magnitude in terms of execution speed and memory footprints.***

Keywords: data mining, frequent patterns, sequential patterns, bitmap presentation

1. Introduction

Agrawal [1] was first to introduce the problem of mining sequential patterns. For a given sequential data, the problem is to find all sequential patterns with a user-defined minimum support, also named frequent sequential patterns. The foremost challenge of mining sequential patterns is the computational cost of support counting for large amount of candidate patterns.

Many research work are carried out in this area and among the related works, after mid's 1990's following Agarwal and Srikant many scholars provided more efficient algorithms [2][3][5][7]. Besides these, work has been done to extend the mining of related patterns. An Apriori-like method finds all frequent items first. By adopting multiple iteration approach, the candidate patterns with length l are generated from the frequent patterns with length (l-1) using iteration. Then the supports of these candidate patterns are checked to discover frequent patterns with length l. The Apriori-like sequential pattern mining methods suffer from the costs to handle a potentially huge set of candidate patterns and scan the database repeatedly. To eliminate the snags present in the afore mentioned algorithms, Prefix Span [4] algorithm developed by Pei et.al, developed from Free Span [6], was proposed and the design of prefix Span was based on divide-and-conquer approach. Recursive method was employed to create sequences where each sequence has the same prefix subsequence. By developing local frequent prefix subsequences in each projected database recursively, all the sequential patterns were discovered without any candidate generation. Although Prefix Span prevented from generating unnecessary candidate patterns, the cost of constructing projected databases recursively was not affordable when dealing with large databases.

In this paper, an improved-version of sequential pattern mining algorithm, called DBP-SPAM, is proposed for mining frequent sequential patterns efficiently. By extending the structures of bitmap data representation, the item presentIn table is constructed first and the corresponding binary position table is constructed next. Based on the binary data presentation, several heuristic mechanisms are proposed to speed up the efficiency of support count and the S-Patterns and I-Patterns are found. Where Support count: min_support is a subtle task: *A too small value may lead to to generation of thousands of patterns, whereas a too big one may lead to no answer found.* To come up with an appropriate min_support, one needs to have prior knowledge about the mining query and the task specific data, and be able to estimate beforehand how man patterns will be generated with a particular threshold. Moreover, the memory footprints required to store temporary data during performing depth-first traversal considerably less than the memory footprints of SPAM algorithm. The experimental results showcased that DBP-SPAM can achieve high magnitude of efficiency outperform the SPAM regarding execution time. Since the proposed algorithm utilizes direct bit position manipulation, the implementation of the algorithm is easy and complexities related to pruning is reduced considerably.

2. Preliminaries

The problem of mining sequential patterns was originally proposed by [1]. The following definitions refer to [1, 3, 5, and 7].

Let I = {i_1, i_2, i_3, … . i_n} be a set of unique items. A sequence S is an ordered list of events, denoted as <e_1, e_2, e_3… en> where e_i is an item, (i.e.) $e_i \in$ I for $1\leq i \leq n$. For brevity, the brackets are omitted if the element has only one element, (i.e.) (a) is written as a. An item can occur multiple times in different event of a sequence. The number of events in a sequence is called the length of a sequence and a sequence of l length is l-sequence. A sequence S_a = {a_1, a_2, a_3.. a_n} is contained in another sequence S_b={b_1,b_2,b_3,.. b_m}, if there exist integers $1 \leq i_1 < i_2 < i_3 \ldots < i_n \leq m$ such that $a_1 = b_{i1}$, $a_2 = b_{i2}$, …. $a_n = b_{in}$. If sequence S_a is contained in another sequence S_b, then S_a is called subsequence of S_b and S_b a super-sequence of S_a, denoted by $S_a \subseteq S_b$.

Table 1: Sample database S

Sid	*Sequences*
S1	< A (CD) AD >
S2	< ACAE >
S3	< CAD(BCD) >
S4	< BBC >
S5	< (BCD)D >

From the table 1, the input sequence database S taken by synthetic data generated by the IBM synthetic market basket data generated. The input sequence database S is a set of tuples (*sid, s*), where *sid* is the sequence identifier and *s* is the input sequence. The number of tuples in S database is called base size of the database S, and denoted as |S|. A tuple (*sid, s*) is said to contain in sequence S_a, If $S_a$is a subsequence of *s*. The support of a sequence $S_a$in the database S is the number of tuples in the database containing S_a, denoted as sup (S_a).

For a given positive integer *min-sup*, as the support threshold, a sequence $S_a$is called frequent sequential pattern in database S, if sup (S_a) ≥*min-sup*. Otherwise the pattern is infrequent.

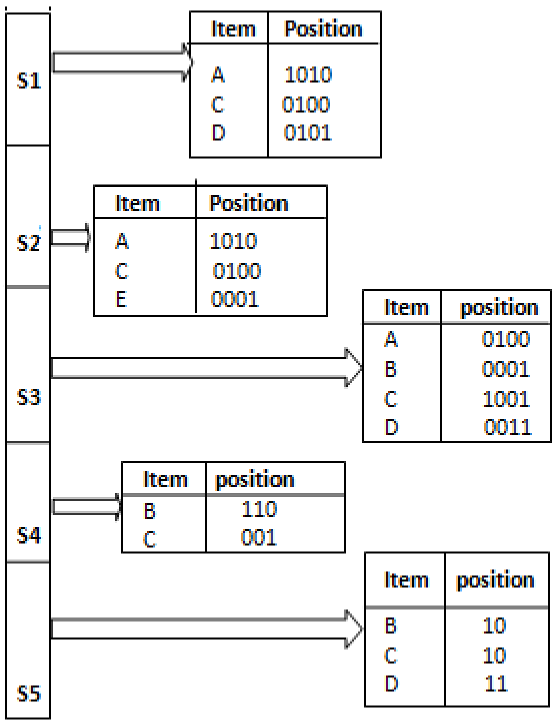
3. Proposed Approach

The proposed approach constructs an item bit position table for each sequence in the database S. Let us consider S1= < A (CD) AD >, here there are four sequences and for brevity, the brackets are omitted and <(A) (CD) (A)(D)> is written as < A (CD) AD >. The item positions are found exploring the sequence left to right and the corresponding positions are stored. The length of the binary represented row in the position table is equal to the length of the sequence in the database S. If the item X is in the ith position of the sequence from left, the ith position of that item X is set to 1, otherwise it is set to 0.

Table 2: Item position in a sequence

< A (CD) AD >				
S1	**A**	**CD**	**A**	**D**
A	1	0	1	0
C	0	1	0	0
D	0	1	0	1

Considering the sequence S1 in the sample database S, there are three elements and four sequences as shown in the table 2. If the item is present in the sequence, it is denoted by 1 else by 0 as shown in the table 2. Since A is present in the sequence 1 and 3, the bit position corresponding to A is 1010.



Item	Position
A	1010
C	0100
D	0101

Item	Position
A	1010
C	0100
E	0001

Item	position
A	0100
B	0001
C	1001
D	0011

Item	position
B	110
C	001

Item	position
B	10
C	10
D	11

Figure 1: Bit Position table of Database S

To reduce the computational cost of checking bits in the position table, Item presence table is constructed with three fields namely Item, Present in and support. Here again we employ top down approach and record the item present in the sequence of database S. If an item is present in ith row of the sequence database, then it is marked by 1, else it is marked by 0. Consider the item “A” in sample sequence database S, since “A” is present in S1, S2, S3 the item presence table is constructed as A = 1, 1, 1, 0, 0. The entire item presentIn table is constructed as shown in the table 3.

Table 3: Item PresentIn table

ITEM	PRESENT IN					SUPPORT
	S1	S2	S3	S4	S5	
A	1	1	1	0	0	3
B	0	0	1	1	1	3
C	1	1	1	1	1	5
D	1	0	1	0	1	3
E	0	1	0	0	0	1

4. DBP-SPAM Algorithm

This section enumerates the working of DBP-SPAM algorithm and the generations of candidates directly from the position and present. In tables are explained clearly. Instead of generating the candidates by inserting a data into a pre-known frequent pattern, the proposed approach directly generates the candidates using the bit position table and present in table. Let us assume that the given min_sup threshold value provided by the user be 2. From the Item presence table, the item “E” is eliminated since the support corresponding to “E” is 1, which is less than 2. (i.e.) sup (E) <min_sup. Now consider the presence table to create candidates, first the item A and B is considered. A = {1 1 1 0 0}, B = {0 0 1 1 1} and here to find (A)(B) and (AB) AND operation is performed in the present in table values of A and B.

Table 4: Checking the presence of A, B

ITEM	*S1*	*S2*	*S3*	*S4*	*S5*
A	1	1	1	0	0
B	0	0	1	1	1
A & B	**0**	**0**	**1**	**0**	**0**

From the table 4, it is found that except S3 no other sequences in the database S contains both A and B. The calculations for S-extended patterns and I-extended patterns are carried out with S3 bit position table.

Table 5: Formation of candidates for A and B

ITEM	*Sequence S3*			
	POSITION			
	1	*2*	*3*	*4*
A	0	1	0	0
B	0	0	0	1

Check the 1's position in both the row and if the 1's position of both A and B are same, then S-extended patterns are formed, (AB). If the position varies, then I-Extended patterns are formed. Since we do not have any 1's in the same position, the S-Extended pattern (AB) = 0, but the I-extended pattern if formed as (A) (B). The support count of (A) (B) = 1, which is less than the given min_sup value of 2, so the candidate is pruned away.

Definition 1: *To form I-extended patterns for items AB, the bit position of A should be definitely less than the bit position of B.*

Bit_A (Pos_i) <Bit_B (Pos_j) => (A) (B), where *i<j*

Definition 2: *To form S-extended patterns for items AB, the bit position of A should exactly match the bit position of B.*

Bit_A (Pos_i) = Bit_B (Pos_j) => (AB) , where *i = j*

Example 1: Let us consider A, C, D
Fetch the values of A = {1 1 1 0 0}, C = {1 1 1 1 1} and D = {1 0 1 0 1} from the presence table and perform AND operation to compute the S-Extended sequences and I-Extended sequences.

Table 6: Checking the presence of A,C,D

ITEM	**S1**	**S2**	**S3**	**S4**	**S5**
A	1	1	1	0	0
C	1	1	1	1	1
D	1	0	1	0	1
A&C&D	**1**	**0**	**1**	**0**	**0**

From the table 6, it is clear that the sequence S1 and S3 contains A, C, D. The candidate formations are carried out with S1 and S3 position tables.

Table 8: Formation of candidates for A, C, D

ITEM	**Sequence S1**			
	POSITION			
	1	**2**	**3**	**4**
A	1	0	1	0
C	0	1	0	0
D	0	1	0	1

ITEM	**Sequence S3**			
	POSITION			
	1	**2**	**3**	**4**
A	0	1	0	0
C	1	0	0	1
D	0	0	1	1

Consider sequence S1, check the 1's position of A, C, and D. According to the *definition 1*,The bit position of A is less than the bit position of C and this satisfies the definition and (A) (C) is formed and the support of (A)(C) is incremented as 1. Again when 1's of D is computed, the bit position of A is less than the bit position of C and the bit position of C is less than the bit position of D. This satisfies the definition 1, and (A)(C)(D) is formed.

(A) (C) =1, (A)(C)(D) =1, (C)(D) = 1
Similarly according to the *definition 1* and *definition 2*, the position of C and D are equal to form I-extended sequence (CD) as it satisfies the defination2, and again the bit position of A is less than the bit position of (CD), the candidate (A)(CD) is formed.
(CD) =1, (A)(CD) =1

Consider sequence S3, The bit position of A is less than the bit position of C and less than D, (A)(C) and (A)(D) support count are incremented. Similarly the bit position of C is less than the bit position of D, (C)(D) support count is incremented.

The candidates formed are (A)(C) = 2, (C)(D) =2

According to *definition 2* and *1*,the bit position value of C and D are equal so the I-extended sequence (CD) is incremented. The bit position value of (CD) is less than the bit position value of (A), and satisfies the *definition1*, (A)(CD) is incremented.

Candidates formed are (CD) = 2, (A)(CD) =2

Since the support count of (A)(C)(D) is less than the min_sup, the candidate is not a frequent sequential pattern and (A)(C)(D) is pruned away. The support count of (A)(CD) is equal to the min_sup threshold and the candidate **(A)(CD)** is a **frequent sequential pattern**. The rest of the 2-itemset candidates which are equal or greater than the min_sup threshold value shown in this example are formed earlier before the 3-itemset sequential pattern formations.

5. Pseudo Code of DBP-SPAM Algorithm

```
ALGORITHM DBP-SPAM ( database D, min-sup)
INPUT: Sequential database D, min-sup
OUTPUT: Sequential patterns
BEGIN:
Initialize bitPositions sequences,
 PresentIn sequences and Scount as zeroes
For each [$sid_i$, s]$\in$ D begin
For each Element $s_j$ of s begin
For each item i$\in s_j$ begin
If $PresentIn_I$ (i) = 0, Mark $PresentIn_I$ (i) = 1
Set $j^{th}$bit in $POS_I$(i) =1
End for
End For
End For
Patterns= IS-patterns(PresentIn, min-sup)
END
```

Figure 2: Pseudo code of DBP-SPAM algorithm

```
FUNCTION IS-Pattern ( Present In table, min-sup)
INPUT: Present In table, min-sup
BEGIN:
For each item i ∈Present In Table
Form base itemsets by applying AND operation
If base Itemsets>= min-sup
Store base itemsets in base table
End for
For each Itemsets K ∈ base table
Fetch POS tables according to the items ∈ K
Find S-Extended patterns based on position
Count the S-extended patterns
If S-extended patterns >= min-sup
Store in Results
Find I-Extended patterns based on equal position
Count I-extended patterns
If I-Extended patterns >=min-sup
Store in Results
End For
Return Results

END
```

Figure 3: Pseudo code of IS-pattern generation

6. Working of IS-Pattern Function

The sample sequential dataset is shown in the table 1 and the presentIn table is constructed as shown in the table 2 and the bit position table is a constructed as shown in figure 1. Let us assume that the user defined *min-sup* value be 2. Here from the table 2, the item "E" is pruned away as the support count value is less than the provided *min-sup* threshold value.

STEP 1:
Fetch each value from the PresentIn table and form the base table.

Table 9: Construction of base table

PATTERN	*AND Operation*	*Result*	*Count*	*PRUNE*
~~AB~~	A=11100 B=00111	AB=00100	Support =1	PRUNED
AC	A=11100 C=11111	AC=11100	Support =3	
AD	A=11100 D=10101	AD=10100	Support = 3	
BC	B=00111 C=11111	BC=00111	Support = 3	
BD	B=00111 D=10101	BD=00101	Support =2	
CD	C=11111 D=10101	CD=10101	Support = 3	
~~ABC~~	AB=00100 C =11111	ABC=00100	Support =1	PRUNED
~~ABD~~	AB=00100 D= 10101	ABD=00100	Support =1	PRUNED
ACD	AC=11100 D = 10101	ACD= 10100	Support =2	
BCD	BC=00111 D =10101	BCD=00101	Support =2	

From the base table shown in the table 9, the sequences with eligible support are AC, AD, BC, BD, CD, ACD and BCD.

STEP 2:
Fetch the newly found sequences from the base table and form I-Extended patterns and S-Extended Pattern.

Consider the first sequence in the base table, AC = 11100. This denotes that the items A and C are present in S1, S2, and S3.
Fetch bit position tables of S1, S2 and S3 to form I-patterns and S-patterns.

ITEM	Sequence S1			
	POSITION			
	1	2	3	4
A	1	0	1	0
C	0	1	0	0

Calculate the bit positions to find the S-Sequence, The positions of A and C are not equal and hence no S-Extended sequences are formed. But I-Extended sequences can be formed since the position of A is less than the position of C. (A)(C) =1, (A) (C) (A) =1

ITEM	Sequence S2			
	POSITION			
	1	2	3	4
A	1	0	1	0
C	0	1	0	0

Here in Sequence S2 the S-Extended patterns are calculated, again the positions of A and C are not equal and no S-Extended patterns are formed. The I-Extended patterns are formed for sequence S2. Since the bit position of A is less than C, the following I-Extended patterns are formed.
(A)(C)=1, (A)(C)(A)=1

ITEM	Sequence S3			
	POSITION			
	1	2	3	4
A	0	1	0	0
C	1	0	0	1

Here in sequence S3, the S-Extended patterns and I-Extended patterns are formed. Since the bit positions of A and C are not equal no S-Extended patterns are formed. But I-Extended patterns are formed as the bit position of A is less than C.

(A)(C) =1

The total count of S-Extended patterns and I-Extended patterns are computed and compared with min-sup value to prune away the infrequent sequential patterns.

Count of (A)(C) =3, (A)(C)(A) =2, both these patterns are either equal or higher than the *min-sup* threshold value given by the user and these two patterns are considered Frequent Sequential patterns.

7. Experimental Evaluation

The proposed DBP-SPAM algorithm was implemented Microsoft Visual Basic 6.0 programming language on a personal computer with 2.66GHz Intel Pentium dual core processor, 1GB RAM running on windows 7 ultimate. The evaluations were performed on synthetic data generated by the IBM synthetic market-basket data generator. The inputted parameters used for comparison are given below in the table 10.

Table 10: Synthetic IBM dataset Parameters

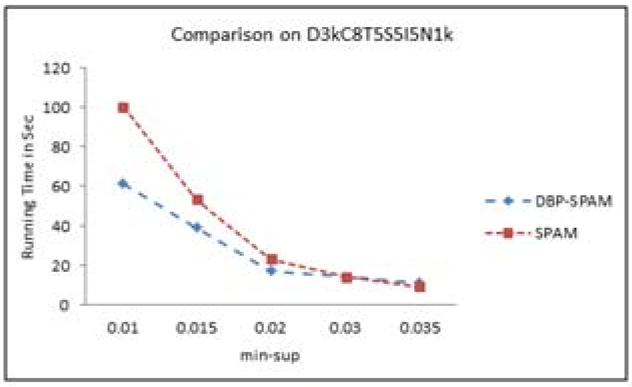
Parameters	*Description of parameter*
D	Total number of sequences in the dataset
C	Average elements per sequence
S	Average length of potentially frequent sequential patterns
I	Average length of itemsets in maximal potentially frequent patterns
T	Average number of items per transactions.
N	Number of different items in 000s.
K	Fixed, which covers the range of typical values

The experimental evaluation regarding running time is compare on different synthetic datasets. These graphs shows the results as the minimum support is changed from 1% to 0.25%. Fig. 4(a), the experiments are carried out with varying *min-sup* values and the proposed algorithm DBP-SPAM showcased that it can outperform the SPAM and attain greater efficiency than SPAM with respect to running time. The foremost reason for this high speed execution relies on the pruning technique, where direct bit position of items is manipulated to form the S-Extended patterns and I-Extended patterns.

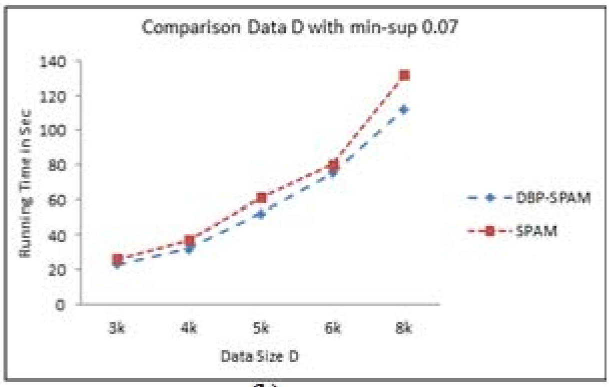
In Fig. 4(b) shows when the *min-sup* value is lower, the DBP-SPAM clearly outperforms the SPAM by a big magnitude and when the *min-sup* value is high, the DBP-SPAM matched the speed of the SPAM on most occasions. On some situations with less number of data sequences (2K), the overheads of calculating the support count of individual items and itemsets increases and thereby reduce the speed of the DBP-SPAM. But on larger datasets like (5K), the proposed DBP-SPAM completely outscores the SPAM. Fig.4 (c) shows as the average number of elements per sequence(C) increase, the size of generated synthetic datasets will increase.

In Fig. 4(d) shows the estimated results on maximal memory usage. As far as the memory footprints are concerned, the proposed algorithm utilizes almost the same memory as the SPAM algorithm when executed and occasionally lesser memory footprints than SPAM. The primary reason for this is eliminating the items and itemsets from the PresentIn and bit position tables to avoid unnecessary storage. In Fig. 4(e) shows the maximal memory requirement of DBP_SPAM and SPAM by varying the average number of elements per sequence in the datasets, which indicates the similar outcome. The results of these tests are shown in table 11 to 13.

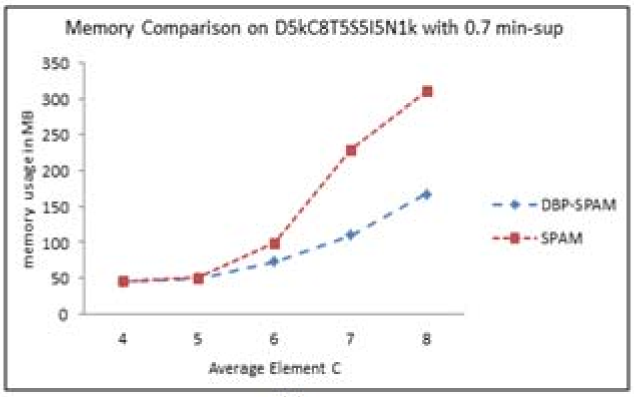
8. Experimental Results



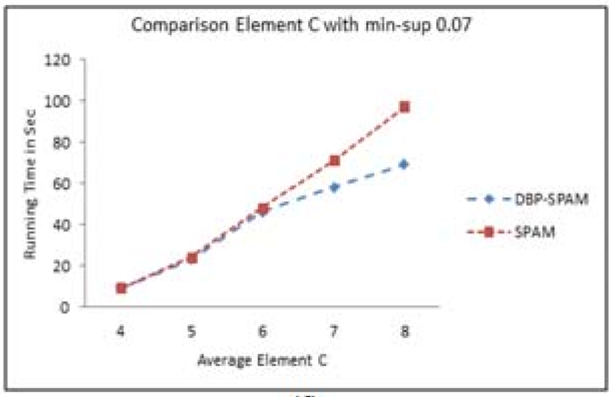
(a)



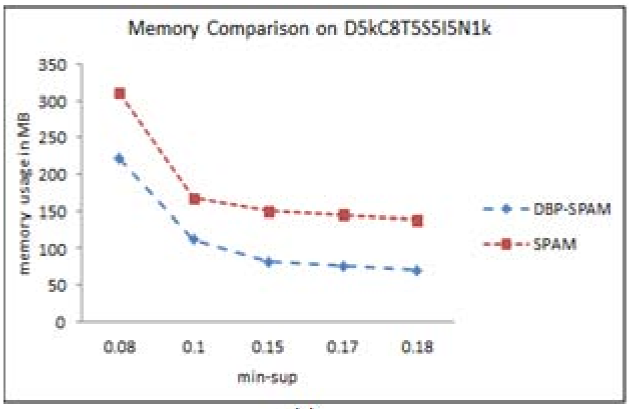
(b)



(c)



(d)



(e)

Figure 4: Experimental Results.

Table 11: Comparison with respect to running time

RUNNING TIME (SEC) - D3KC8T5S5I5N1K					
ALGORITHM	MIN-SUP				
	0.01	0.015	0.2	0.03	0.035
SPAM	99	58	26	19	12
DBP-SPAM	62	41	22	18	11
	DATA SIZE D				
	4K	5K	6K	7K	8K
SPAM	21	33	58	76	132
DBP-SPAM	20	31	43	63	109
	AVERAGE ELEMENT C				
	4	5	6	7	8
SPAM	9.2	23	43	71	93
DBP-SPAM	9	22	40	54	66

Table 12: Comparison w.r.t memory

MEMORY IN MB– D5KC8T5S5I5N1K					
ALGORITHM	MIN-SUP				
	0.08	0.1	0.15	0.17	0.18
SPAM	312	168	162	160	157
DBP-SPAM	213	117	79	76	70
	AVERAGE ELEMENT C				
	4	5	6	7	8
SPAM	45.2	46.4	75	212	317
DBP-SPAM	46	46	62	81	191

Frequent sequential Patterns formed for dataset S

Table 13: Output sequential patterns

Patterns	*Count*	*Patterns*	*Count*
AA	2	BCD	2
AC	3	BD	2
ACA	2	CA	3
A(CD)	2	CAD	2
AD	2	CD	3
ADD	2	(CD)D	2
BC	2		

9. Conclusion

A new algorithm to mine sequential patterns is proposed in this paper and the bit position manipulation approach used in the proposed algorithm reduced the unnecessary checking and speed up the execution considerably. The foremost challenge in sequential pattern mining depends on confining the size of the candidates generated and abridging the computations involved for the support count. The memory footprints needed to store the data is much less or equal to the existing SPAM algorithm. The experimental results showcased that the proposed algorithm outperformed the SPAM on larger datasets and smaller *min-sup* threshold values.

References

[1] R. Agarwal and S.Arya, .Mining multiple level Association Rules to mining Multiple level Correlation to discover complex patterns. In Proc. 2012, International Journal of Computer Science, 2012.
[2] Alpa Reshamwala and Dr. Sunita Mahajan, Improving Efficiency of Apriori Algorithms for sequential Pattern mining. International Journal of data mining, In March 2014.
[3] KMVM Kumar, PVS Srinivas, CR Rao – MS-SPADE: Sequential Pattern Mining with multiple minimum supports. In Proc. 2012, International Journal of Computer science 2012.
[4] HJ Shyur, C Jou, K Chang: A data mining approach to discovering reliable sequential patterns, Journal of Systems and software, 2013.
[5] Manan Parik, Barat Caudari and chetna Chand: A comparative Study of Sequential Pattern Mining Algorithms, International Journal of Application or Innovation in Engineering & Management. Vol 2, Feb 2013.
[6] J. Pei, J. Han, Pattern Growth Methods: Frequent Pattern Mining. In Proc. 2014 Springer.
[7] S.MuthuSelvan, Ks. Sundaram, "A survey of Sequence Patterns in Data Mining Techniques. International Journal of Applied Engineering Research 2015.