Sequential Pattern Mining: A Comparison between GSP, SPADE and Prefix SPAN

[1]Manika Verma, [2]Dr. Devarshi Mehta
[1]Assistant Professor, [2]Associate Professor
[1]Department of Computer Science, Kadi Sarva Vishwavidyalaya,
[2]GLS Institute of Computer Technology, Ahmedabad, India

***Abstract* - This paper presents a comparison between basically three kinds of algorithm GSP (Generalized Sequential Pattern), SPADE (An efficient Algorithm for mining Frequent Sequences) and Prefix Span (Prefix-projected Sequential Pattern Mining). GSP is the Apriori based Horizontal formatting method, SPADE is the Apriori based vertical formatting method and Prefix-SPAN is Projection-based pattern growth method. This paper elaborate step wise explanation of each algorithm demonstrating number of iterations required in each algorithm. Later a comparison is made between Total time required to execute algorithm, count of frequent sequences found and Max memory (in mb) required by algorithms GSP, SPADE and Prefix-SPAN. The above stated attributes i.e. total time; frequent sequences and Max Memory are obtained using SPMF (A sequential Pattern Mining Framework).**

***Keywords* - GSP, SPADE, Prefix-Span, Apriori-based, Projection-based**

I. INTRODUCTION

A sequential pattern is a series of item-sets; item-sets in sequences are in specific order.Sequential pattern mining helps to extract the sequences which are most frequent in the sequence database, which in turn can be interpreted as domain knowledge for several purposes[6]. Sequential pattern mining is used in various areas for different purposes. It can be used for identifying Customer Shopping Sequence [8][7]. Various algorithms have been implemented for identifying frequent sequences from sequence database. One of the approaches used to identify frequent sequences is apriori approach. "The apriori approach is based on the apriori property, as introduced in the context of association rule mining. This property states that if a pattern a is not frequent then any pattern bthat contains acannot be frequent. Two of the most successful algorithm that takes this approach are GSP and SPADE. The major difference between the two is that GSP uses a horizontal data format while SPADE uses vertical one. The sequence growth approach does not require candidate generation; it gradually grows the frequent sequences. Prefix span which originated from FP-growth, uses this approach as follows: first it finds the frequent single items, then it generates a set of projected database, one database for each frequent item. Each of these databases is then mined recursively while concatenating the frequent single item, then it generates a set of projected databases, one database for each frequent item. Each of these database is then mined recursively while concatenating the frequent single items into a frequent sequence. These algorithms perform well in database consisting of short frequent sequences. However, when mining databases consisting of long frequent sequences e.g. stocks values, DNA chains, or machine monitoring Data, their overall performance worsens by an order of magnitude." [8]

Original Input Sequence DataBase[1]

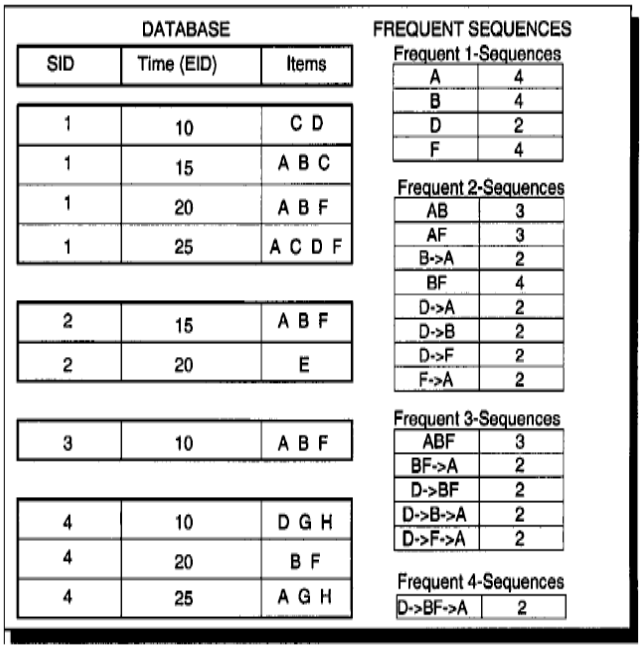

Figure 1. Original input-sequence database.

Figure 1

To demonstrate step by step explanation of the algorithms GSP, SPADE and Prefixspan the above database shown in Figure 1 is used as input. For all the algorithms, minimum support is 2. The Candidate sets are denoted by C. Ckspecifies candidate sets having k items. For Eg C2 mean Candidate sets having 2 items. The Candidate sets that satisfy minimum support belongs to Lk. In Lk ,Kdenotes number of items in sequence.

II. ALGORITHMS OF SEQUENTIAL PATTERN MINING

A. *Generalized Sequential Pattern (GSP) Algorithm*[2]

GSP discovers sequential pattern. GSP scales linearly with the number of data sequences, and has very good scale-up properties with respect to the average data sequence size. [2]

Step by step explanation of how does GSP works:

The database in figure 1 is taken in following format to demonstrate step by step explanation of GSP.

Table 1.1

SID	**TIME(EID)**	**ITEMS**
1	10,15,20,25	<(CD)(ABC)(ABF)(ACDF)>
2	15,20	<(ABF)(E)>
3	10	<(ABF)>
4	10,20,25	<(DGH)(BF)(AGH)>

Table 1.1 To determine frequent items using GSP, consider minimum support as 2 in this example.

Table 1.2

Items	No. of occurrence of Items
A	4
B	4
C	1
D	2
E	1
F	4
G	1
H	1

The items that satisfy minimum support are, A,B,D,F . So A,B,D,F are Frequent 1- Sequence

1. Frequent 1- itemsets are A,B,D,F
2. Using Apriori property, frequent itemsets obtained are shown in table below. C is used for denoting Candidate sets (Item sets). Ck denotes candidates having k items. For eg C1 denotes item sets (or candidate sets) having 1 item.

Table 1.3 **C1**

Items	No. of occurrence of Items
A	4
B	4
C	1
D	2
E	1
F	4
G	1
H	1

The items that satisfies minimum support belongs to L1. Lk denotes candidate having k items. For eg L1 denotes items sets (or candidate sets) having 1 item.

Table 1.4 **L1**

Items	No. of occurrence of Items
A	4
B	4
D	2
F	4

3. From above Table 1.4 we can see that a,b,c,d are the that satisfy minimum support.
   C2 (where k=2) can be obtained by joining L1 X L1 but the condition is k-2 items must be common, so here while joining for obtaining items sets for C2 (2-2=0) items must be common

Table 1.5 **C2**

Items	No. of occurrence of Items
A->A	1
A->B	1
A->D	1
A->F	1
AB	3

AD	1
AF	3
B->A	2
B->B	1
B->D	1
B->F	1
BD	1
BF	4
D->A	2
D->B	2
D->F	2
D->D	1
DF	1
F->A	2
F->B	1
F->D	1
F->F	1

Candidates that satisfy minimum support belongs to L2

Table 1.6 **L2**

Items	**No. of occurrence of Items**
AB	3
AF	3
B->A	2
BF	4
D->A	2
D->B	2
D->F	2
F->A	2

4. Now, in this step we have to achieve 3-sequence item-sets, i.e item-sets having 3 items. C3(where k=3) can be obtained by joining L2 X L2 but the condition is k-2 items must be common, so here while joining for obtaining items sets for C3 (3-2=1) items must be common.

Table 1.7 **C3**

Items	**No. of occurrence of Items**
ABF	3
AB->A	1
AF->A	1
BF->A	2
D->B->A	2
D->F->A	2
D->BF	2
F->AF	1
D->FA	1
D->AB	1
F->AB	0

B->AB	1
B->AF	1

Candidates having minimum support belong to L3

Table 1.8 **L3**

Items	**No. of occurrence of Items**
ABF	3
BF->A	2
D->B->A	2
D->F->A	2
D->BF	2

5. In this step we have to achieve 4-sequence item-sets, i.e item-sets having 4 items. C4(where k=4) can be obtained by joining L3 X L3 but the condition is k-2 items must be common, so here while joining for obtaining itemsets for C4 (4-2=2) items must be common.

Table 1.9 **C4**

Items	**No. of occurrence of Items**
D->BF->A	2

Candidates having minimum support qualify for moving into L4.

Table 1.10 **L4**

Items	**No. of occurrence of Items**
D->BF->A	2

Here, we have obtained only 1, 4-sequence(Sequence of item set having 4 items) item set.

B. *SPADE(An efficient Algorithm for mining Frequent Sequences)*[1]

SPADE is the algorithm used for fast discovery of Sequential pattern . Most of the sequential pattern mining algorithms assume horizontal database layout .**Spade** uses vertical database format, where we maintain a disk based id-list for each item, show in following figure. [1]

A		B		D		F	
SID	EID	SID	EID	SID	EID	SID	EID
1	15	1	15	1	10	1	20
1	20	1	20	1	25	1	25
1	25	2	15	4	10	2	15
2	15	3	10			3	10
3	10	4	20			4	20
4	25						

Figure 2 ID-List for ATOMS

To determine frequent items using SPADE, consider minimum support as 2 in this example.

Consider the database shown in figure 1 above. For generating 1-sequence (single items) the occurrence of each item is being counted thus those items that satisfy the minimum support (here minimum support=2) belongs to L1.1-Sequence items are specified in above figure. Above figure specifies Id-List for the atoms.

Step by step explanation of how does SPADE works:

1. Items and count of their occurrence is given in table below

Table 2.1 **C1**

Items	**No. of occurrence of Items**
A	4
B	4
C	1
D	2
E	1
F	4
G	1
H	1

Table 2.2 **L1**

Items	**No. of occurrence of Items**
A	4
B	4
D	2
F	4

2. Following steps shows how using temporal join we can identify frequent items.
   Here we wish to compute the support of sequence AB. The set is J= {A,B,D,F}. We can perform ***temporal join*** one at a time to obtain final-id list as shown in figure below. Here the relationship between AB is non-temporal one, which we call equality join, since they must occur at the same time. We thus find all occurrences of A and B with the same e-id and store them in id-list shown below.[7,page 40]

Table 2.3 Id-list for AB (Non-temporal join)

AB		
SID	EID A	EID B
1	15	15
1	20	20
2	15	15
3	10	10

In above table, SID is same mean the sequence occurred two times but belongs to same customer, so in this case we ignore 1 count for sequence SID=1, so total number of occurrence of AB is 3. Thus minimum support is satisfied.

Table 2.4 Id-list for AD (Non-temporal join)

AD		
SID	EID A	EID D
1	25	25

For AD, the number of occurrence is 1, which does not satisfy minimum support.

Table 2.5 Id-list for AF (Non-temporal join)

AF			
SID	EID	A	EID F
1	20		20
1	25		25
2	15		15
3	10		10

For AF, the number of occurrence is 3, which satisfies minimum support. While counting number of occurrences here two sequences are for same SID-1, so one count is ignored and hence total number of occurrence is 3.

Table 2.6 Id-list for BD (Non-temporal join)

BD			
SID	EID	B	EID D
-	-		-

For BD, the number of occurrence is 0, which does not satisfy minimum support.

Table 2.7 Id-list for BF (Non-temporal join)

BF			
SID	EID	B	EID F
1	20		20
2	15		15
3	10		10
4	20		20

For BF, the number of occurrence is 4, which satisfies min support.

Table 2.8 Id-list for DF (Non-temporal join)

DF			
SID	EID	D	EID F
1	25		25

For DF, the number of occurrence is 1, which does not satisfy minimum support.

Table 2.9 Id-list for A->A (Temporal join)

A->A			
SID	EID	A	EID A
1	15		20
1	20		25

For A->A, the number of occurrence is 2, but both occurrence belongs to same SID so number of occurrence is actually 1, which does not satisfy minimum support.

Table 2.10 Id-list for A->B (Temporal join)

A->B			
SID	EID	A	EID B
1	15		20

For A->B, the number of occurrence is 1, which does not satisfy minimum support.

Table 2.11 Id-list for A->D (Temporal join)

A->D		
SID	EID A	EID D
1	15	25
1	20	25

For A->D, the number of occurrence is 1 and not 2 because both occurrences are for same SID so we would ignore one count and occurrence of A->D is 1 which does not satisfy minimum support.

Table 2.12 Id-list for A->D (Temporal join)

A->F		
SID	EID A	EID F
1	15	20
1	15	25
1	20	25

For A->F, the number of occurrence is 1 and not 3 because all the three occurrences are for same SID so we would ignore 2 counts and occurrence of A->F is 1 which does not satisfy minimum support.

Table 2.13 Id-list for B->A (Temporal join)

B->A		
SID	EID B	EID A
1	15	20
4	20	25

For B->A, the number of occurrence is 2, which satisfies the minimum support.

Table 2.14 Id-list for B->B (Temporal join)

B->B		
SID	EID B	EID D
1	15	15
1	20	20

For B->B, the number of occurrence is 2, but both occurrence belongs to same SID so number of occurrence is actually 1, which does not satisfy minimum support.

Table 2.15 Id-list for B->D (Temporal join)

B->D		
SID	EID B	EID D
1	15	25
1	20	25

For B->D, the number of occurrence is 1 and not 2 because both the occurrences are for same SID so we would ignore 1 count and occurrence of B->D is 1 which does not satisfy minimum support.

Table 2.16 Id-list for B->F (Temporal join)

B->F		
SID	EID B	EID F
1	20	25

For B->F, the number of occurrence is 1 which does not satisfy minimum support.

Table 2.17 Id-list for D->D (Temporal join)

D->D			
SID	EID	D	EID D
1	10		25

For D->D, the number of occurrence is 1 which does not satisfy minimum support.

Table 2.18 Id-list for D->A (Temporal join)

D->A			
SID	EID	D	EID A
1	10		20
1	10		25
4	10		25

For D->A, the number of occurrence is 3, but 2 occurrences are for same SID so number of occurrence for D->A is 2, which satisfies the minimum support.

Table 2.19 Id-list for D->B (Temporal join)

D->B			
SID	EID	F	EID D
1	10		15
1	10		20
4	10		20

For D->B, the number of occurrence is 2, which satisfies the minimum support.

Table 2.20 Id-list for D->F (Temporal join)

D->F			
SID	EID	D	EID F
1	10		20
1	10		25
4	10		20

For D->F, the number of occurrence is 3, but two occurrences are for same SID, so the number of occurrence of D->F is 2 which satisfy the minimum support.

Table 2.21 Id-list for F->F (Temporal join)

F->F			
SID	EID	F	EID F
1	20		25

For F->F, the number of occurrence is 1, which does not satisfy the minimum support.

Table 2.22 Id-list for F->D (Temporal join)

F->D			
SID	EID	F	EID D
1	20		25

For F->D, the number of occurrence is 1, which does not satisfy the minimum support.

Table 2.23 Id-list for F->B (Temporal join)

F->B			
SID	EID	F	EID B
1	20		25

For F->B, the number of occurrence is 1, which does not satisfy the minimum support.

Table 2.24 Id-list for F->A (Temporal join)

F->A		
SID	EID F	EID A
1	20	25
4	20	25

For F->A, the number of occurrence is 2, which satisfies the minimum support.

From non-temporal and temporal joins obtained above, below is the table showing count of occurrence of sequences of items.

C2

Items	No. of occurrence
AB	3
AD	1
AF	3
BD	0
BF	4
DF	1
A->A	1
A->B	1
A->D	1
A->F	1
B->A	2
B->B	1
B->D	1
B->F	1
D->D	1
D->A	2
D->B	2
D->F	2
F->F	1
F->D	1
F->B	1
F->A	1

L2 (Sequences that satisfy minimum support in C2)

Items	No. Of occurrence
AB	3
AF	3
BF	4
B->A	2
D->A	2
D->B	2
D->F	2
F->A	2

3. This step generates 3-sequence

Table 3.1 list for ABF (Non-Temporal join)

ABF			
SID	EID A	EID B	EID F
1	20	20	20
2	15	15	15
3	10	10	10

For ABF, the number of occurrence is 3, which satisfies the minimum support.

Table 3.2 Id-list for AB->A (Temporal join)

AB->A			
SID	EID A	EID B	EID A
1	15	15	20
1	20	20	25
2	15	15	-
3	10	10	-

For AB->A, the number of occurrence is 2, but as both sequence belongs to SID-1which does not satisfy the minimum support. In this case number of occurrence is 2 because we have sequence AB->A for SID 1, but for SID 2 and 3 we have AB together but AB is not followed by A, thus – (dash) is specified.

Table 3.3 Id-list for AF->A (Temporal join)

AF->A			
SID	EID A	EID F	EID A
1	20	20	25
1	25	25	-
2	15	15	-
3	10	10	-

For AF->A, the number of occurrence is 1, which does not satisfy the minimum support.

Table 3.4 Id-list for BF->A (Temporal join)

BF->A			
SID	EID B	EID F	EID A
1	20	20	25
2	15	15	-
3	10	10	-
4	20	20	25

For BF->A, the number of occurrence is 2, which satisfies the minimum support.

Table 3.5 Id-list for D->B->A (Temporal join)

D->B->A			
SID	EID D	EID B	EID A
1	10	15	20
4	10	20	25

For D->B->A, the number of occurrence is 2, which satisfies the minimum support.

Table 3.6 Id-list for D->F->A (Temporal join)

D->F->A			
SID	EID D	EID F	EID A
1	10	20	25
4	10	20	25

For D->F->A, the number of occurrence is 2, which satisfies the minimum support.

Table 3.7 Id-list for D->BF (Temporal join)

D->BF			
SID	EID D	EID B	EID F
1	10	20	20
2	-	15	15
3	-	10	10
4	10	20	20

For D->BF, the number of occurrence is 2, which satisfies the minimum support.

Table 3.8 Id-list for F->AF (Temporal join)

F->AF			
SID	EID F	EID A	EID F
1	-	20	20
1	-	25	25
2	-	15	15
3	-	10	10

For F->AF, the number of occurrence is 0, which does not satisfy the minimum support.

Table 3.9 Id-list for D->AF (Temporal join)

D->AF			
SID	EID D	EID A	EID F
1	10	20	20
1	10	25	25
2	-	15	15
3	-	10	10

For D->AF, the number of occurrence is 2, but both occurrences belongs to same SID-1, so actually number of occurrence is 1, which does not satisfy the minimum support.

Table 3.10 Id-list for D->AB (Temporal join)

D->AB			
SID	EID D	EID A	EID B
1	10	15	15
1	10	20	20
2	-	15	15
3	-	10	10

For D->AB, the number of occurrence is 2, but both occurrence belongs to same SID-1, so actually number of occurrence is 1, which does not satisfy the minimum support.

Table 3.11 Id-list for F->AB (Temporal join)

F->AB

SID	EID F	EID A	EID B
1	-	15	15
1	-	20	20
2	-	15	15
3	-	10	10

For F->AB, the number of occurrence is 0, which does not satisfy the minimum support.

Table 3.12 Id-list for B->AB (Temporal join)

B->AB			
SID	EID B	EID A	EID B
1	-	15	15
1	15	20	20
2	-	15	15
3	-	10	10

For B->AB, the number of occurrence is 1, which does not satisfy the minimum support.

Table 3.13 Id-list for B->AF (Temporal join)

B->AF			
SID	EID B	EID A	EID F
1	15	20	20
1	15	25	25
1	20	25	25
2	-	15	15
3	-	10	10

For B->AF, the number of occurrence is 3, but all the three sequences belongs to same SID, so number of occurrence is 1,which does not satisfy the minimum support.

From temporal joins obtained above, below is the table showing count of occurrence of sequences of items.

C3

Items	No. of occurrence
ABF	3
AB->A	1
AF->A	1
BF->A	2
D->B->A	2
D->F->A	2
D->BF	2
F->AF	0
D->AF	1
D->AB	1
F->AB	0
B->AB	1
B->AF	1

L3

Items	No. of occurrence
ABF	3
BF->A	2
D->B->A	2
D->F->A	2
D->BF	2

4. This step generates 3-sequence

Table 4.1 Id-list for ABF->A (Temporal join)

ABF->A				
SID	EID A	EID B	EID F	EID A
1	20	20	20	25
2	15	15	15	-
3	10	10	10	-

For ABF->A, the number of occurrence is 1, which does not satisfy the minimum support.

Table 4.2Id-list for D->BF->A (Temporal join)

D->BF->A				
SID	EID D	EID B	EID F	EID A
1	10	20	20	25
2	-	15	15	-
3	-	10	10	-
4	10	20	20	25

For D->BF->A, the number of occurrence is 2, which satisfy minimum support.

C. *Prefix-projected Sequential Pattern Mining (PREFIX SPAN) Algorithm*[3]

PrefixSpan explores prefix-projection in sequential pattern mining. PrefixSpan mines the complete set of patterns but greatly reduces the efforts of candidate sequence generation.Prefix-projection substantially reduces the size of projected databases and leads to efficient processing. [3]

From the figure 1 of database, following given table is taken.

SID	**TIME(EID)**	**ITEMS**
1	10,15,20,25	<(CD)(ABC)(ABF)(ACDF)>
2	15,20	<(ABF)(E)>
3	10	<(ABF)>
4	10,20,25	<(DGH)(BF)(AGH)>

Figure 3

For below given example, consider minimum support as 2.

Step by step explanation of how does SPADE works:

1. Count number of occurrence for each itema

Items	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
No.of frequent items	4	4	1	2	1	4	1	1

The items that satisfy minimum support are, A,B,D,F. So A,B,D,F are Frequent 1- Sequence.

2. Divide Search Space

<A>	<B>	<D>	<F>
(_BC)(ABF)(ACDF)	(_BC)(ABF)(ACDF)	(ABC)(ABF)(ACDF)	(ACDF)
(_BF)(E)	(_F)(E)		(E)
(_BF)	(_F)		
(_GH)	(_F)(AGH)	(_GH)(BF)(AGH)	(AGH)

3. Find Subsets

3.1- Subsets for <A>

<A>
(_BC)(ABF)(ACDF)
(_BF)(E)
(_BF)
(_GH)

3.1.1<A>

A	B	F	C	D	E	_B	_C	_F	_G	_H
1	1	1	1	1	1	3	1	2	1	1

The items that satisfy minimum support are shown in table below

_B	_F
3	2

<AB>	<AF>
(_C)(ABF)(ACDF)	(ACDF)
(_F)E	E
(_F)	

The items Frequent 2-Sequence are AB and AF, as they comes under 2 sequences and thus satisfy minimum support.

Considering prefix AB for (_C)(ABF)(ACDF) for <AB>

<AB>
(_C)(ABF)(ACDF)
(_F)E
(_F)

<AB>

A	B	C	D	E	_F	F	_C
1	1	1	1	1	2	1	1

_F
2

<ABF>
(ACDF)
E

The items Frequent 3-Sequence is ABF, as it comes under 2 sequence and thus satisfy minimum support.

Considering prefix AB for sequence (_F)(ACDF)

<AB>
(_F)(ACDF)
(_F)E
(_F)

A	**C**	**D**	**F**	**E**	**_F**
1	1	1	1	1	3

<ABF>

<ABF>
(ACDF)
E

The items Frequent 3-Sequence is ABF, as it comes under 2 sequence and thus satisfy minimum support. Same is being detected in above example

<AF>

A	**C**	**D**	**E**	**F**
1	1	1	1	1

No items frequent.

3.2 –Subsets for <B>

<B>
(_BC)(ABF)(ACDF)
(_F)(E)
(_F)
(_F)(AGH)

3.2.1<B>

A	**_B**	**C**	**D**	**E**	**_F**	**G**	**B**	**F**	**H**	**_C**
2	1	1	1	1	3	1	1	1	1	1

Items that satisfy minimum support i.e. 2 are A, F

A	**_F**
2	3

Considering prefix B->A for sequence (_BF)(ACDF)

B->A

(_BF)(ACDF)
(_GH)

B->A is frequent as it comes under two sequences

A	**_B**	**C**	**D**	**_F**	**_G**	**_H**	**F**
1	1	1	1	1	1	1	1

From above table no more items are frequent

Considering prefix B->A for sequence (_CDF)

B->A

(_CDF)
(_GH)

B->A is frequent as it comes under two sequence

_C	**_D**	**_F**	**_G**	**_H**
1	1	1	1	1

From above table no more items are frequent

BF

(ACDF)
E
AGH

BF is frequent as it comes under two sequences

A	**C**	**D**	**E**	**F**	**G**	**H**
2	1	1	1	1	1	1

BF->A

(_CDF)
(_GH)

BF-> A is frequent as it occurs in two sequences.

_C	**_D**	**_F**	**_G**	**_H**
1	1	1	1	1

From above table no more items are frequent

3.1.3 Find Subsets

<D>
(ABC)(ABF)(ACDF)
(_GH)(BF)(AGH)

A	**B**	**C**	**D**	**F**	**_G**	**G**	**_H**	**H**
2	2	1	1	2	1	1	1	1

D->A

(_BC)(ABF)(ACDF)
(_GH)

D->A is frequent as it comes under 2 sequences.

A	**B**	**_B**	**C**	**_C**	**D**	**F**	**_G**	**_H**
1	1	1	1	1	1	1	1	1

No item satisfy min support

Considering prefix D->B for for sequence (_C)(ABF)(ACDF)

D->B

(_C)(ABF)(ACDF)
(_F)(AGH)

D->B is frequent as it comes under 2 sequences.

A	B	C	_C	D	F	_F	G	H
2	1	1	1	1	1	2	1	1

D->B->A

(_BF)(ACDF)
(_GH)

D->B ->A is frequent as it comes under 2 sequences.

A	_B	C	D	F	_G	H	_F	_H
1	1	1	1	1	1	1	1	1

No more item from above table satisfy min support

Considering prefix D->B for for sequence (_F)(ACDF)

D->B

(_F)(ACDF)
(_F)(AGH)

_F	A	C	D	F	G	H
2	2	1	1	1	1	1

D->BF

ACDF
AGH

D->BF is frequent as it comes under two sequences

A	C	D	F	G	H
2	1	1	1	1	1

D->BF-A

(_CDF)
(_GH)

D->BF->A is frequent as it comes under two sequences

_C	_D	_F	_G	_H
1	1	1	1	1

No more item from above table satisfy min support

D->B->A

(_BF)(ACDF)
(_GH)

D->B->A is frequent.

A	C	D	F	_G	_H	_B	_F
1	1	1	1	1	1	1	1

D->F

(ACDF)
(AGH)

D-> F comes under 2 sequences so it is frequent

A	C	D	F	G	H

2	1	1	1	1	1

D->F->A

(_CDF)
(_GH)

D-> F->A comes under 2 02sequences so it is frequent

_C	**_D**	**_F**	**_G**	**_H**
1	1	1	1	1

No more item from above table satisfy min support

3.1.4 Find Subsets

<F>
(ACDF)
(E)
(AGH)

A	**C**	**D**	**E**	**F**	**G**	**H**
2	1	1	1	1	1	1

F->A

(_CDF)
(_GH)

F->A is frequent as it comes under 2 sequences so it is frequent

_C	**_D**	**_F**	**_G**	**_H**
1	1	1	1	1

No more item from above table satisfy min support.

III. COMPARISON BETWEEN GSP,SPADE AND PREFIXSPAN

	GSP	**SPADE**	**PrefixSpan**
Purpose	GSP algorithm is used for extracting frequently occurring sequences and is proposed by Agrawal and Srikant[2]	SPADE algorithm is used for fast (faster than GSP) discovery of sequential pattern and is proposed by MOHAMMED J. ZAKI[1]	PrefixSpan algorithm is used for discovery of sequential pattern and performs better in mining large sequences.It is proposed by Jian Pei et.al[3]
PPerformance	GSP is an iterative algorithm. It scans the database number of times depending on the length of the longest frequent sequences in database. The I/O cost is substantial (large) if database contains very long frequent sequences. [2]	Spade outperforms the GSP by a factor two, and by an order of magnitude with precomputed support of 2-sequences. It has excellent scaleup properties with respect to a number of parameters such as number of input-sequences, the number of events per-input sequence, the event size and the size of potential maximal frequent events and sequences. [1]	PrefixSpan mines the complete set of patterns but greatly reduces the efforts of candidate sequence generation.prefix-projection substantially reduces the size of projected Databases and leads to efficient processing.[3]. A comprehensive performance study shows that PrefixSpan, in most cases, outperforms the a priori-based algorithm GSP, FreeSpan,

			and SPADE[5]
Format of database	Uses Horizontal Format Database[2]	Uses Vertical Format Database [1]	Uses Horizontal Format Database [3]
Approach	Apriori Based[2]	Apriori Based[1]	Prefix-Projecte Pattern-Growth Based[3]
Candidate Sequence	Candidate Sequence generation required[2]	Candidate Sequence generation required.[1]	Candidate Sequence generation is not required. It uses prefix projection.[3]

IV. ANAYSIS

Using the SPMF (A Sequential Pattern mining framework) [9] algorithms, the above mentioned three algorithms GSP,SPADE and PrefixSpan were executed on the same dataset. The result of analysis is displayed below:
The Table below shows the comparison among algorithms. Algorithms were executed on a text file named ContextSpade. Size of the file was 145 bytes.

Attributes	GSP	SPADE	PrefixSpan (Max Pattern Length 4)
Total Time	~ 140 ms	~ 78 ms	~ 15ms
Frequent Sequences Count	3917	3917	700
Max Memory(mb)	8.184585571289062	8.397941589355469	6.2912750244140625700

The following chart shows comparison among attributes of GSP,SPADE and PrefixSpan

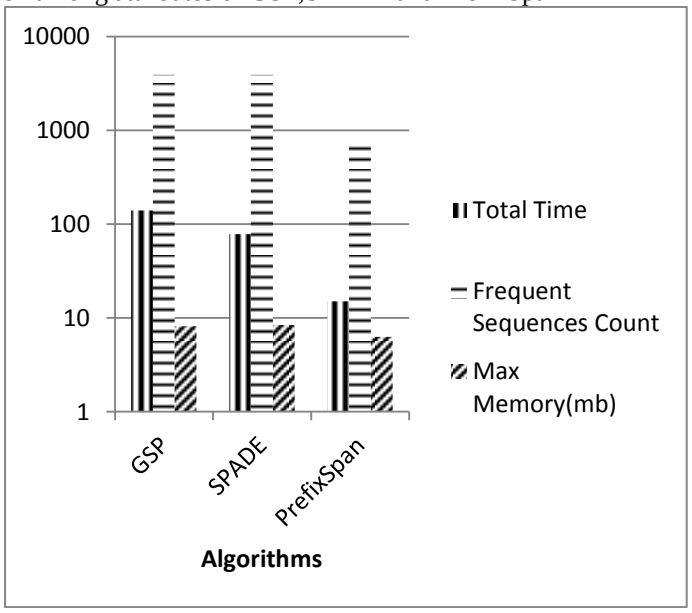

Figure 3

V. CONCLUSION

This paper made the comparison among GSP, SPADE and PrefixSpan algorithms. This paper presented the concept and explanation of the above mentioned algorithms. Using SPMF it is determined that PrefixSpan is better than GSP and SPADE in performance as it takes lesser time than GSP and SPADE. SPADE takes less time than GSP but it takes quite more time than PrefixSpan. The memory required by PrefixSpan is much less than GSP and SPADE while the memory required by GSP and SPADE is same. Various attributes like Total Time required for executing algorithm, frequent sequences found by algorithms and Maximum memory used by algorithm

REFERENCES

[1] Mohammed J Zaki. (2001). SPADE: An efficient algorithm for Mining Frequent Sequences.In Journal Machine Learning, Volume 42 Issue 1-2, January-February 2001 Pages 31-60.

[2] Ramakrishna Srikant, Rakesh Agarwal (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. In proceedings of the 5th International Conference on EDT:Advances in Database Technology Pages 3-17

[3] Jian Pei Jiawei Han Behzad Mortazavi-Asl Helen Pinto, Qiming Chen Umeshwar Dayal Mei-Chun Hsu.(2001).PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, , published in Data Engineering, 2001. Proceedings. 17th International Conference on , Page(s):215 - 224

[4] A GSP based efficient algorithm for mining frequent sequences, Minghua Zang, Ben Kao, Chi-Lap Yip, David Cheung

[5] Jian Pei, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen,Umeshwar Dayal. (2004). Mining Sequential Patterns by Pattern-Growth:The PrefixSpan Approach. In IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 16, NO. 10, OCTOBER 2004

[6] ThabetSlimani and Amor Lazzez.(2013). SEQUENTIAL MINING:PATTERNS AND ALGORITHMS ANALYSIS.In International Journal of Computer and Electronics Research [Volume 2, Issue 5, October 2013]

[7] R. Srikant et.al. Data Mining:Concepts and Techniques Mining sequence patterns in transactional databases. http://www.cs.nyu.edu/courses/spring08/G22.3033-003/8timeseries.ppt

[8] Yaron Gonen,Nurit Gal-Oz,Ran Yahalom, and Ehud Gudes. (2010). CAMLS: A constraint based Apriori Algorithm for mining Long Sequences. In proceedings of Database Systems for Advanced Applications, 15th International Conference

[9] SPMF: A Sequential Pattern Mining FrameWork http://www.philippe-fournier-viger.com/spmf/