Sequential Pattern Mining With Multiple Minimum Supports in Progressive Databases

K.M.V.Madan Kumar[1], P.V.S.Srinivas[2] and C.Raghavendra Rao[3]

[1]Research Scholar, CMJ University, Shillong, Meghalaya, India
Assoc Professor, TKR College of Engg &Tech, Hyderabad, AP, India
[2]Professor, Geethanjali College of Engineering & Technology, Hyderabad, India
[3]Professor, Dept. of CIS, University of Hyderabad, Hyderabad, India

Abstract

Although there may be lot of research work done on sequential pattern mining in static, incremental, progressive databases, the previous work do not fully concentrating on support issues. Most of the previous approaches set a single minimum support threshold for all the items or item sets. But in real world applications different items may have different support threshold to describe whether a given item or item set is a frequent item set. This means each item will contain its own support threshold depends upon various issues like cost of item, environmental factors etc. In this work we proposed a new approach which can be applied on any algorithm independent of that whether the particular algorithm may or may not use the process of generating the candidate sets for identifying the frequent item sets. The proposed algorithm will use the concept of "percentage of participation" instead of occurrence frequency for every possible combination of items or item sets. The concept of percentage of participation will be calculated based on the minimum support threshold for each item set. Our algorithm MS DirApp, which stands for Multiple Support Direct Appending, which discovers sequential patterns in by considering different multiple minimum support threshold values for every possible combinations of item or item sets.

Keywords

Multiple minimum supports, Progressive sequential pattern, Percentage of participation

1. INTRODUCTION

Sequential Pattern mining is one of the most important research issues in data mining. Which was first introduced by Agarwal and Srikanth [2] and can be described as follows: we are given a set of data sequences which will be used as input data. Each sequence consists of a list of ordered item sets containing a set of different items. The sequential pattern mining finds all Sub sequences with frequencies lower than min support which is given by user. After that there have been a significant research work was done by various researchers on sequential pattern mining and defined number of algorithms not only for static data base [2],[3] Where data do not change over time but also for incremental data bases[6] where there will be new data arriving as the time goes by. In addition to above researchers some other researchers derived other kind of sequential patterns like closed sequential patterns [7], [8], [9]. Constraint sequential pattern [10] maximal

DOI: 10.5121/ijdms.2012.4403

sequential patterns [11] spatio temporal sequential patterns [12]. Sequential pattern on specific type of data [13] on stream data [14].

The sequential patterns mining in progressive data bases will be having a vide use in many applications. For example in mobile applications, share market. Moreover the stock price changes of the company before five years ago may have very little influence on the quotes of other stocks. To remedy the problem of sequential pattern mining in progressive databases, Haung.et.al [6] proposed an algorithm DirApp which takes the concept of period of Interest (POI) but Haung.et.al[6] also considered only uniform min.sup like all the previous researchers.

But considering uniform min.sup, implicitly assumes that all items in the data base have similar frequency. However some items may appear very frequently in the data base while other rarely appears. Under such circumstances, if we set the value of min support too high we will not find those rules involving rare items in the data base. On the other hand if we set that value too low, it will generate huge amounts of meaningless patterns. Therefore liue .et.al [4] first address this and propose the concept of multiple minimum supports (MMS in Short)in association rule mining. In this study we first extend the definition of sequential pattern by considering the concept of MMS, which allows users to specify multiple min.sup for each item. Items with low frequencies will be specified with lower minimum support and pattern involving these items can be easily retrieved for further decision support.

But the major problem on frequent pattern mining with MMS is that the downward closure property no longer holds in the mining process, which means that a super-pattern of an infrequent pattern might be frequent pattern. To effectively reduce the search space in a level –wise methods, Liu.et.al proposes the sorted closure property, where all item in data sets are sorted in ascending order by their MIS values. The sorted closure property however is invalid in sequential pattern mining since the order in the data sequences cannot be altered. Therefore to discover complete set of sequential patterns with MMS is not straight forward. Based on the new definitions of sequential patterns with MMS, we first proposed the concept called Percentage of participation (POP) where POP is the percentage of participation of the item or item set with respect to the no of sequences in the POI i.e. |Db| and minimum item support (MIS).

The structure of this paper as follows. In section 2, we first review the concept of DirApp as the basis of our approach. Section 3 introduces the definition of the progressive sequential pattern mining with multiple minimum supports (MMS) by involving the Percentage of participation (POP).Our algorithm called MS-DirApp will be discussed in section 4 and we have conclusion in section 5.

2. RELATED WORK

Haung.et.al proposed algorithm which works for mining of sequential patterns in progressive data bases. The input of progressive sequential pattern mining is a user specified length of POI and user-defined minimum support threshold. POI is a sliding window protocol whose length is a user specified time interval continuously advances as the time advances. The sequences having elements whose time stamps are within the POI will be considered to contribute for |Db| for current sequential patterns (where |Db| is the number of sequences in the POI). The sequences having elements with time stamps older than the POI will be considered as absolute and removed. In this they consider an element which is set of items appeared in a sequence with the time stamp

denoted by p, q. When the sequence database is denoted by Db then a sequence in a sequence database is a set of elements ordered by time stamp increasingly. They also introduced a time interval denoted by [p, q] to represent the time interval between time stamp p and q. Therefore $Db^{p,q}$ is defined as a subset of the databases Db containing the elements of sequences from time stamp p to time stamp q. They also aimed finding the sequential patterns containing no repeated elements and having more than one element. In the beginning, the mining algorithm receives new data of different timestamps with a user-defined minimum support threshold and the length of POI. For each POI, the mining algorithm progressively updates the sequences in the database along with the occurrence frequencies of candidate sequential patterns. Then, the algorithm outputs frequent sequential patterns which are qualified by the uniform minimum support threshold. Note that the mining algorithm should prune away obsolete data from the sequence database, delete obsolete sequential patterns rapidly, and update |Db| with the number of sequences which have elements in the current POI. The process will be continuously executed until there is no more newly arriving data.

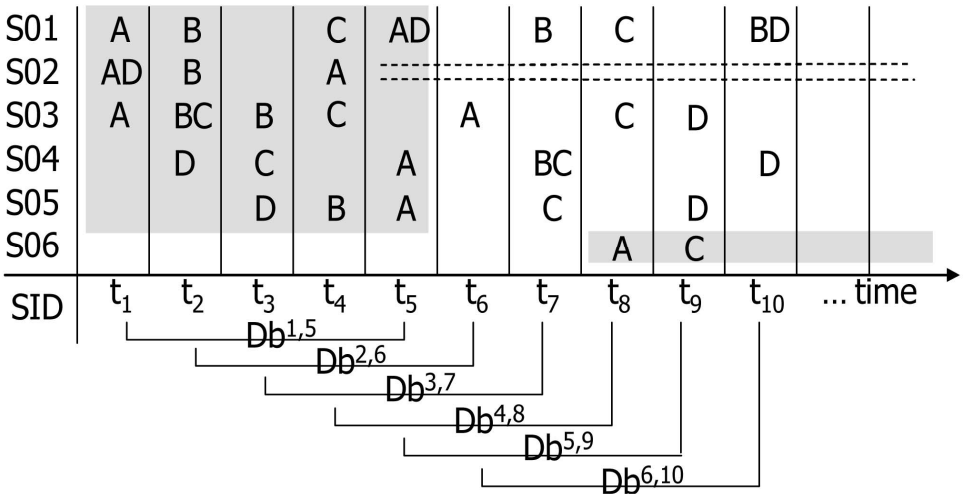

Figure 1

For the above example shown in figure1 the algorithm DirApp constructs the candidate sets as follows which is shown in figure2.

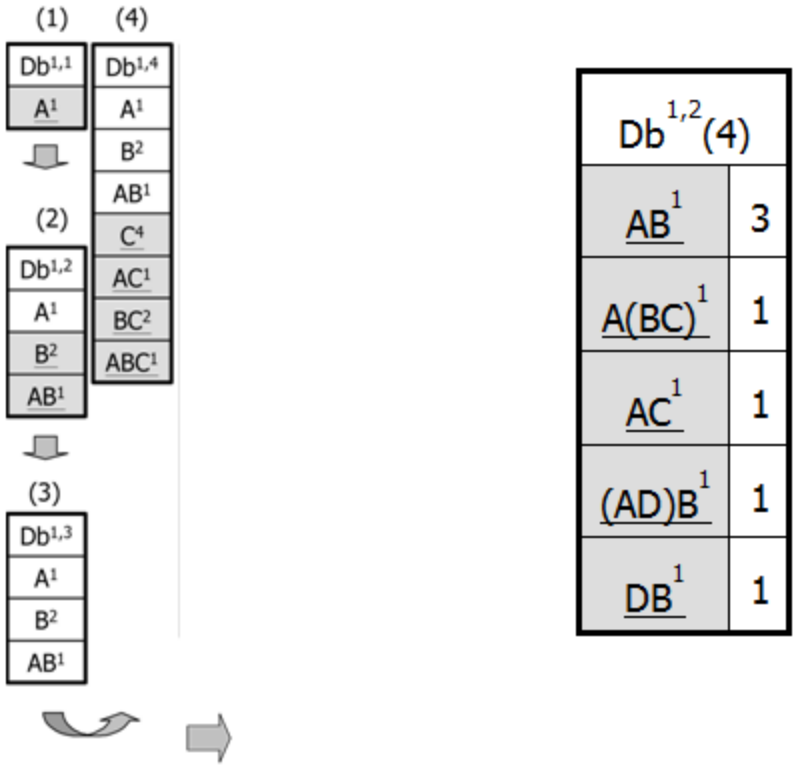

Figure 2

Before this many researchers have developed various methods to find frequent sequential patterns with a static database. Apriori All [2], GSP [3] are the milestones of sequential pattern mining algorithms based on traditional association rules mining technique. SPADE [5], illustrated by Zaki, systematically searched the sequence lattice spanned by the subsequence relation. Han et al. and Pie et al. brought up Free Span and Prefix Span, which found sequential patterns by constructing sub databases of the entire database. Zhang et al. proposed two algorithms GSP+ and MFS+ based on static algorithms GSP and MFS (also derived by the same authors).But as discussed early in section1 all the researchers considered uniform min.sup to describe whether the sequence is frequent or not.

3. PROBLEM DEFINITION

In this section, we formally give definitions used in the discovery of sequential pattern with MMS. Let I denote the set of items in the database, and a subset of I is called an item set. A customer's data-sequence is an ordered list of items with time stamps. Therefore, a sequence, say α, can be represented as $<(a_1: t_1), (a_2: t_2), (a_3: t_3), \ldots, (a_n: t_n)>$, where a_j is an item, and t_j stands for the time when a_j occurs, $1 \leq j \leq n$, and $tj-1 \leq tj$ for $2 \leq j \leq n$.

If several items occur at the same time in t_j for 2 j the sequence, they are ordered alphabetically.

Definition 1. Given an item set $I_q = (i_1, i_2...i_m)$, we say item set I_q occurs in α if integers $1 \leq k_1 < k_2 < \ldots < k_m \leq n$ exist such that, $i_1 = a_{k1}$, $i_2 = a_{k2}$,... ,$i_m = a_{km}$ and $t_{k1} = t_{k2} = ... = t_{km}$. We refer to k_1 and t_{k1} jointly as the position and the time that I_q occurs in α, respectively.

Definition 2. Let $\beta = <I_1, I_2...I_s>$ (I_q subset or equal to I for $1 \leq q \leq s$) be a sequence of item set. Assume that each I_q in β occurs in α. Then we say sequence β occurs in α, or is a subsequence of α if $t_{I1} < t_{I2} < ... < t_{Is}$, where t_{Iq} ($1 \leq q \leq s$) is the time, at which I_q occurs in α.

Definition 3. A sequence database S is formed by a set of records $<sid, s>$, where *sid* is the identifier of this data-sequence and s is a data-sequence. For a given sequence β, the support count of sequence β in S are defined as follows:

supp $(\beta) = |\{(sid, s)|\ (sid, s) \in S \wedge \beta$ *is a subsequence in* $s\}|$

The following definitions are related to the concept of MMS. In this model, the definition of the minimum support is changed. Each item in the database can have its *minsup*, which is expressed in terms of *minimum item support* (MIS). In other words, users can specify different MIS values for different items.

Definition 4: Let MIS (i) denote the MIS value of item i (i subset or equal to I). Given an item set $I_q = (i_1,i_2,...,i_m)$, the MIS value of item set $I_q = (i_1,i_2,...,i_m)$ ($1 \leq k \leq m$), denoted as MIS(I_q), is equal to:

$$\min [\text{MIS}(i_1), \text{MIS}(i_2), \ldots, \text{MIS}(i_m)]$$

Definition 5: Given a sequence $\beta = <I_1 I_2...I_s>$
(I_q subset or equal to I for $1 \leq q \leq s$), the minimum support threshold of β, denoted as MIS (β), is equal to:

$$\min [\text{MIS}(I_1), \text{MIS}(I_2), \ldots, \text{MIS}(I_s)]$$

Definition 6: Given a sequence database S and a sequence β, we call β is frequent in S or β is a sequential pattern in S if *supp* (β) ≥MIS (β).

Definition 7: Progressive sequential pattern mining problem. "Given a user-specified length of POI and a user- defined minimum support threshold, find the complete set of frequent subsequences whose occurrence frequencies are greater than or equal to the minimum support times the number of sequences in the recent POI of a progressive database."

Definition 8: Let I be an item or item set and MIS (I) be the minimum item support threshold of item I. $|Db^{p,q}|$ is the number of sequences In the POI. Timestamps between p ,q.

$$\text{Percentage Of Participation(POP)} = 100/ |Db^{p,q}| * \text{MIS(I)}$$

With the provision of different minimum item support thresholds for different items, user can effectively express different support requirements for different data-sequences. MMS allow users to have higher minimum supports for sequences involving high frequent items, and also allows us to have lower minimum supports for sequences involving rare items. Given a sequence Database S and a set of MIS values for all items in S, we discover all sequential patterns that satisfy MIS (β).

4. THE PROPOSED ALGORITHM

4.1 Candidate sequential pattern generation

The basic idea of MS-DirApp is progressively update each sequence in the flat structure and accumulate the percentage of participation of the candidate sequential pattern from one POI to other POI. In the process of accumulating the candidate sets if the POI is advancing for one timestamp, generally the items belongs to first time stamp in previous POI will be withdrawn along with the percentage of participation as the obsolete data. When new element (item or itemset) are arriving from the progressive time stamp MS-DirApp creates all combinations of items as candidate elements and accumulate the percentage of participation by calculating the POP of the element. In calculation process MS-Dir App consider the least minimum support of the item, if no of items are contributed in the combination. For example if the arriving element is (ABC) the combinations of candidate element are A, B, C, (AB), (AC), (BC) and (ABC) and in the above cases of (AB) it will consider the least minimum support of A or B. This is will be continued with the remaining elements like (AC),(BC) and (ABC).Then MS-DirApp finds the corresponding candidate set of the sequence according to the sequence ID of the element. In the process of forming new candidate sequential patterns with corresponding begin timestamps, MS-DirApp joins every combination of the newly arriving data to the candidates which have already been accumulated in the previous candidate set. The timestamp of the begin element will be considered as the begin timestamp when a candidate sequential pattern appear in a sequence of progressive database. After that MS-DirApp inserts the newly formed candidate sequential patterns into the existing candidate sequential pattern. In addition to that MS-DirApp also calculates the Percentage of participation of the combined elements and accumulates POP of all candidate sequential patterns in another candidate set. For those candidate sequential patterns with obsolete begin timestamp (Begin is smaller than current time minus POI), MS-DirApp deletes them from candidate sets along with their percentage of participation. Finally MS-DirApp outputs frequent sequential patterns based on the multiple minimum support of each item and prunes away obsolete candidate sequential patterns for all sequences.

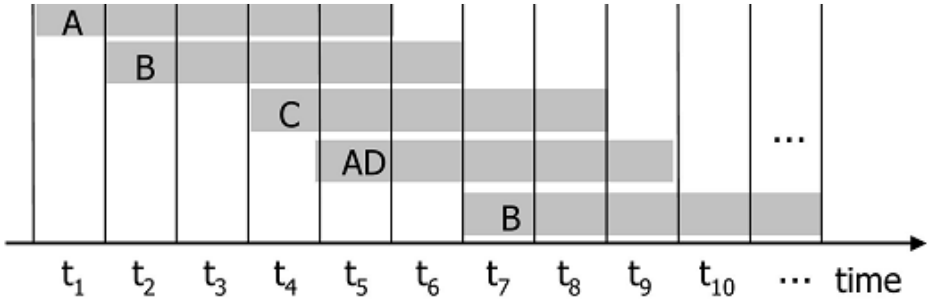

Figure 3

Now in this example we will discuss how the MS-DirApp will generate the candidate sequential pattern and frequent sequential patterns in a progressive databases with multiple minimum supports.

Algorithm POP (Items, MIS(I))

1. var POP;
2. var I, I_{min};// Items
3. For (all combinations of *items* in the *ele*);
4. If (*No of items* >1);
5. Check and *identify* the item I_{min} having less support;
6. POP (*ele*)=100/|$Db^{p,q}$|*MIS(I_{min}): // |$Db^{p,q}$| is no of sequences from p to q
7. **else**
8. POP (*ele*)=100/|$Db^{p,q}$|*MIS(I): // |$Db^{p,q}$| is no of sequences from p to q

END

Figure 4(a)

Algorithm MS-DirApp (POI)

1. var *PT;* // Progressive Table
2. var *current Time*; // timestamp now
3. var *eleSet*; // used to store elements *ele*
4. **while** (there is still new transaction)
5. *eleSet*=read all *ele* at *current Time*;
6. **ck** (*currentTime, PT*);
7. *current Time*++

END

Figure 4(b)

Algorithm ck (*current time, PT*)

1. for (each *ck* of ***PT*** in post order)
2. **if** (*ck* is null)
3. **for** (*ele* of every *seq* in *eleSet*)
4. **for** (all combination of elements in the *ele*)
5. **if** *(element*==label of one of *ele.ck*)
6. **if** (*seq* is in *subck.seq_list*)
7. update timestamp of *seq* to *currentTime, POP (ele);*
8. **else**
9. create a new *sequence* with *current Time*, *POP(ele);*
10. **else**
11. create a new *subck* with *element, current Time, POP(ele);*
12. **else for** (every *seq* in the *seq_list*)
13. **if** (*seq.timestamp*<= *currentTime_POI*)
14. delete *seq*, *POP (el)* from *seq_list* and continue to *next seq*;
15. **if** (there is new *ele* of the *seq* in *eleSet*)
16. **for** (all combination of elements in the *ele*)

```
18. if (element is not in ck)
19. if (element==label of one of subck)
20. if (seq is in subck.seq_list)
21. subck.seq_list.seq.timestamp= seq.timestamp;
22. else
23. create a new sequence with seq.timestamp, POP (ele);
24. else
25. create a new subck with element, seq.timestamp, POP (ele);
28. if (seq_list.Total POP (el)>=100%)
29. output the ele as a sequential pattern;

END
```

Figure 4(c)

The input database is the same data base which was mentioned in Figure1.But here we are considering those elements that sequence S01 has with their corresponding existing POI as shown in figure 3 and the MIS values for A,B,C,D are 0.8,0.5,0.3,0.2 continuously . The algorithms for percentage of participation is shown in figure 4(a) and MS-DirApp is shown in figure 4(b, c). The candidate set of candidate sequential patterns for S01maintained by MS DirApp is shown in figure 5. The $Db^{p, q}$ at the top of each table in figure5 represents the database containing elements along with their percentage of participation from timestamp p to timestamp q. First, MS DirApp reads the element A at timestamp 1 and stores A as a candidate with begin =1, i.e., A^1, in the candidate set along with its percentage of participation. The percentage of participation is calculated basing on the timestamp and the user defined minimum value of support. For example if the user defined minimum support for A is 0.8, the percentage of participation of element A for its each contribution will be 100/ (number of sequence in $Db^{1, 1}$ (3)*MIS of A (0.8)) i.e. 41.66 % as per the definition 8. When receiving element B at timestamp 2, for the candidate patterns already in the candidate set, MS DirApp appends B to A^1 to be AB^1 and accumulates the percentage of participation as shown in the second table of figure 5. Here for the combination of AB, we will consider the minimum value of MIS between A, B. For example the user specified minimum item supports of A, B are 0.8, 0.5 consecutively, we will consider the minimum value i.e. 0.5 for the combination AB. Now, the POP of combination AB will be 100/ (number of sequence in $Db^{1, 2}$ (4)*MIS of B (0.5)) i.e. 50% for its each occurrence as per the definition 8. This process will be continued in MS Dir App, accumulates B^2 and its percentage of participation along with the AB^1.The second table in figure 5, shows this process.

The same process continues until there are no new arriving elements. When MS DirApp deals with$Db^{1, 5}$ at timestamp 5, the newly arriving element is AD, combinations of this element are A, D, and AD. Because there is already an A^1 in the candidate set, MS DirApp changes it's begin timestamp from 1 to 5 and just accumulates its percentage of participation by calculating it. The reason is that for all elements whose timestamps are between 2 and 4 are already appended to A^1. Therefore, we have to only append the elements to A5 whose timestamps are larger than 5. In addition, MS DirApp also appends A to the candidates that do not contain an A in the candidate set along with their POP. Thus, BA^2 with POP 40%, CA^4 with POP 66%, and BCA^2 with POP 66% are inserted into the candidate set. After five timestamps, the POI moves to time interval [2, 6] and accumulates the combinations along with the percentage of participation as shown in figure.5 table 6.

The algorithm not only accumulates newly arrived candidate patterns and also removes obsolete patterns along with percentage of participations i.e. whose time stamps begin with 1 as shown in figure 5 table 6. When MS DirApp handles $Db^{3,7}$ the candidate patterns whose begin time stamps are less than 3 can be deleted too.

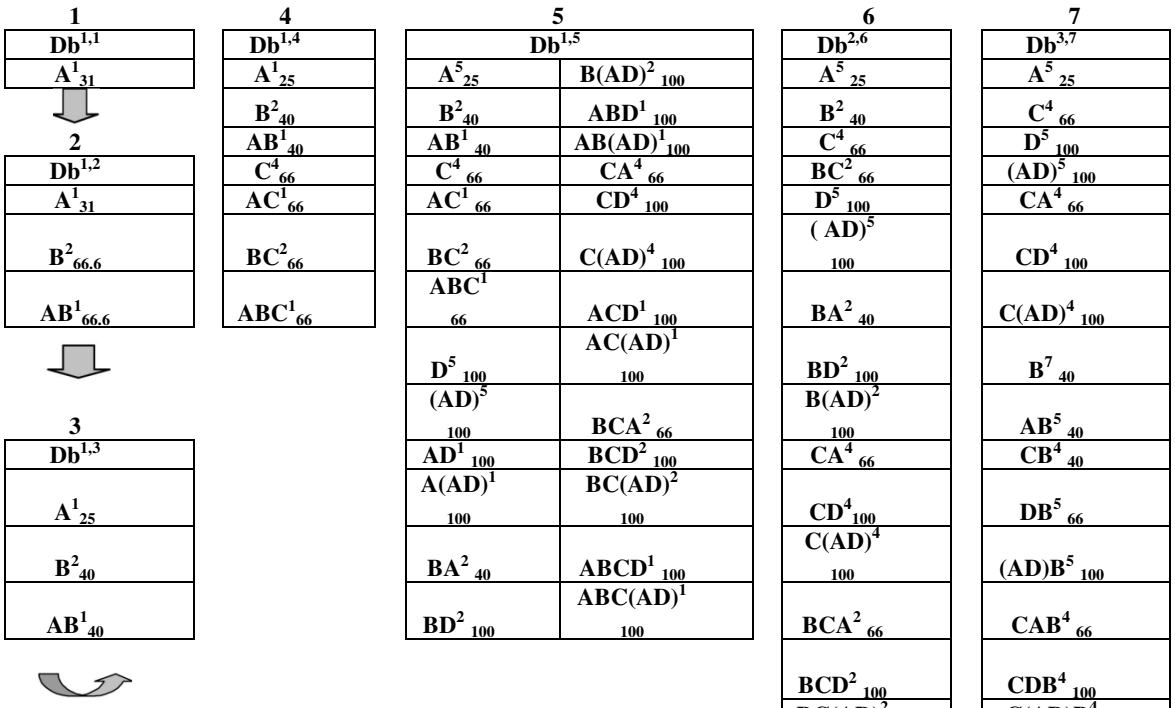

Figure 5

4.2 Frequent sequential pattern generation

Now in this section we will discuss how the frequent sequential patterns will be generated from the candidate sequential patterns. Here we should consider the candidate sequential patterns for all the sequences (S01 to S06) shown in fig1. The process of construction of candidate sequential patterns for the first sequence was discussed previously in section 4.1.The same procedure will be adapted for all the sequences mentioned in figure 1.The next process is identifying the candidate sequential patterns with respect to the time stamps say for example $Db^{1,2}$, $Db^{1,3}$, $Db^{1,4}$, $Db^{1,5}$, $Db^{2,6}$, $Db^{3,7}$ for all the sequences and accumulate the total percentage of participation of the sequences accordingly.

S01
$Db^{1,2}$
A^1_{31}
$B^2_{66.6}$
$AB^1_{66.6}$

(1)

S02
$Db^{1,2}$
A^1_{31}
D^1_{125}
$(AD)^1_{125}$
B^2_{50}
$AB^1_{66.6}$
DB^1_{125}
$(AD)B^1_{125}$

(2)

S03
$Db^{1,2}$
A^1_{31}
B^2_{50}
C^2_{83}
$(BC)^2_{83}$
$AB^1_{66.6}$
AC^1_{83}
$A(BC)^1_{83}$

(3)

S04
$Db^{1,2}$
D^2_{125}

(4)

$DB^{1,2}$ (4)
AB^1_{50}
$A(BC)^1_{83}$
AC^1_{83}
$AD(B)^1_{125}$
DB^1_{125}

(5)

Figure 6

In the process of frequent sequential pattern mining we generally will not consider the sequences who are directly participating without any combinations as frequent sequences. For example $\mathbf{A^1}$ and $\mathbf{B^2}$ in table1, $\mathbf{A^1}$,$\mathbf{D^1}$,$\mathbf{(AD)^1}$ and $\mathbf{B^2}$ in table 2,$\mathbf{A^1}$,$\mathbf{B^2}$,$\mathbf{C^2}$ and $\mathbf{(BC)^2}$ in table 3,$\mathbf{D^2}$ in table 4 of figure 6 will not accumulated for frequent pattern generation. Finally we will treat those sequences whose total percentage of participation should be equal to or more than100 are frequent sequences. The process is shown in the figure 6 and the frequent sequential patterns for $Db^{1,2}$ are $\mathbf{AB^1}$, $\mathbf{AD\ (B)^1}$, $\mathbf{Db^1}$.

5.Result

Our proposed algorithm is very effectively working for test dataset and we have analysed the test dataset for different parameters like period of interest, execution time etc.

5.1 Impact of period of interest (POI) on execution time

Execution time is the time required to execute all the instructions in the proposed algorithm. Here we can observe that the execution time will increase a little bit with respect to the time taken for identifying the item having minimum MIS value. For this we applied the quick sorting technique .It can be noted that execution time is directly proportional to POI. The reason is that, the increase in POI required more time for process as the no of new elements will be added to the existing one.

Figure 7 shows the impact of POI over the execution time.

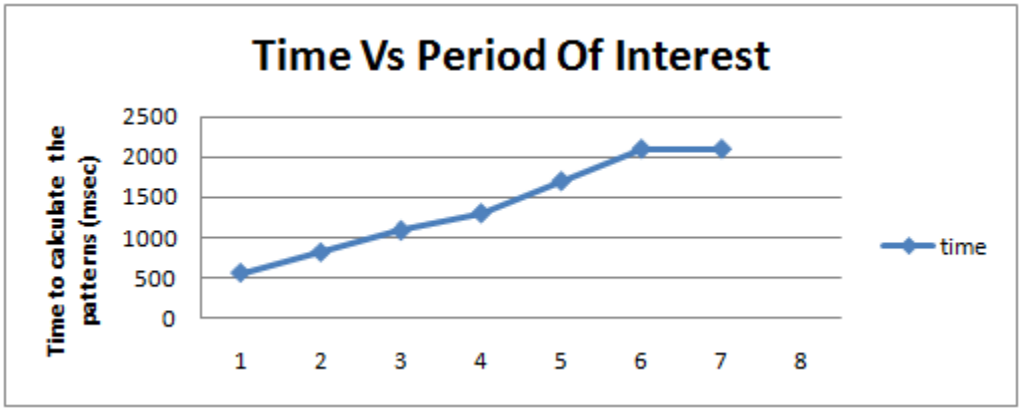

Figure 7

5.2 Impact of POI over number of patterns

Number of patterns is dependent on period of interest a. As from Fig.8 we can see that as the period of interest increases, the number of patterns also increases. This is because as the period of interest increases, the algorithm has more items to process and so they give more number of patterns.

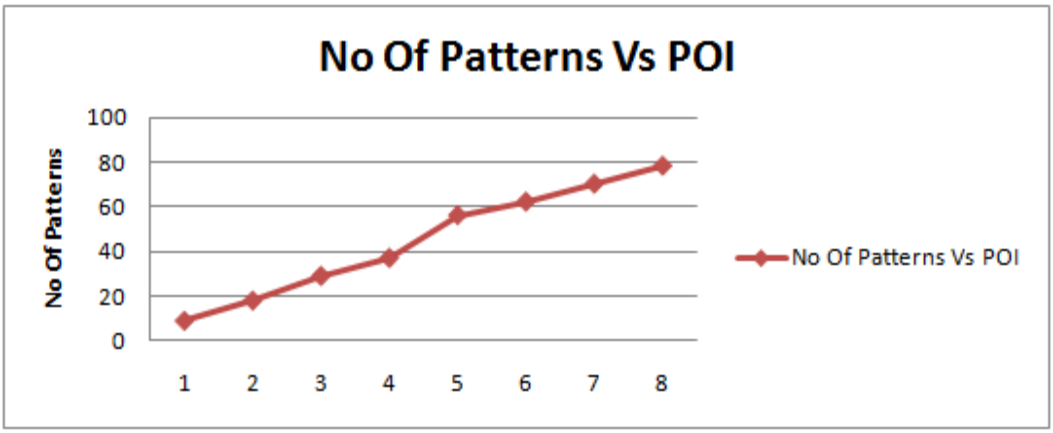

Figure 8

4.3 Comparison between MS-DirApp and DirApp in terms of time in progressive database

Generally mining frequent sequential patterns with multiple minimum supports will take more time when compared with mining frequent sequences with uniform support. In case of percentage of participation (POP), the time for execution of Ms-**DirApp** will take little bit more time than the DirApp . The difference of time between these two will be the time for identifying the item having minimum MIS value from the newly arriving element and accumulating the POP. We can observe the difference in the Figure9

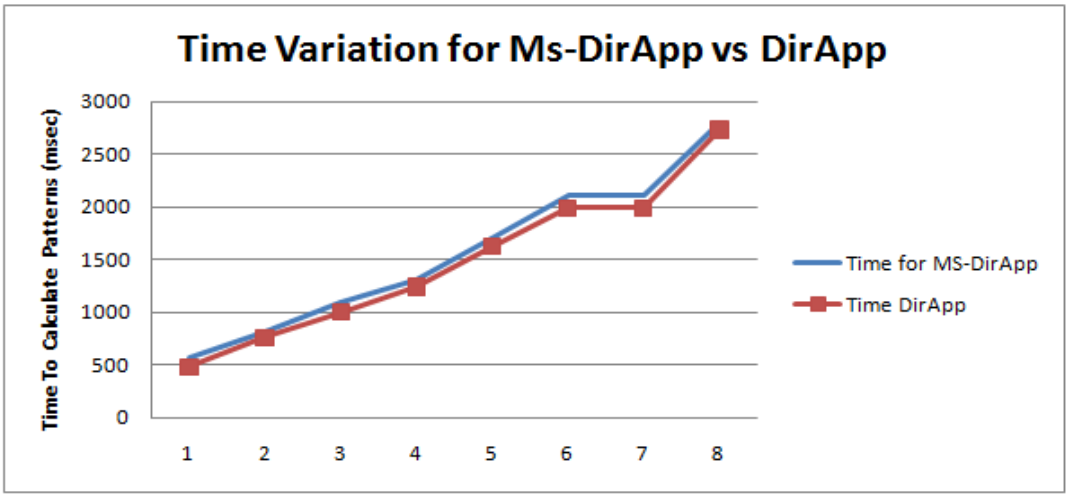

Figure 9

6. CONCLUSION

In the proposed novel work MS DirApp algorithm is efficiently working by using the concept of percentage of participation (POP) for mining frequent sequences in progressive databases with multiple minimum supports. Even though the algorithm is taking little more time for execution when compared with DirApp (which finds the frequent sequential patterns in progressive databases by considering uniform min.support for different items) it works on frequent sequences with multiple minimum supports for different items. As the novel algorithm is finding the sequences with multiple minimum supports we can ignore the little increase in time of execution. But in order to calculate the POP of all candidate sequential patterns, it keeps all the candidate sets for all sequences in every POI. This involves huge memory usage and involves lots of computation.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases (VLDB '94), pp. 478-499, Sept. 1994.

[2] R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. 11th Int'l Conf. Data Eng. (ICDE '95), pp. 3-14, Feb. 1995.

[3] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," Proc. Fifth Int'l Conf. Extending Database Technology (EDBT '96), Mar. 1996.

[4] B.Liu, W.Hsu and Y.Ma, "Mining association rules with multiple minimum supports", Proceedings of the fifth ACM, SIGKDD conference Sandiego,CA,USA,August 15-18,1999,P.341

[5] M.J. Zaki, "Efficient Enumeration of Frequent Sequences," Proc. Seventh ACM Int'l Conf. Information and Knowledge Management (CIKM '98), Nov. 1998.

[6] Jen-Wei Huang, Chi-Yao Tseng, Jian-Chih Ou, and Ming-Syan Chen, Fellow, IEEE: A General Model for Sequential Pattern Mining with a Progressive Database IEEE Transactions On Knowledge and Data Engineering VOL. 20, NO. 9, SEPTEMBER 2008

[7] S Cong, J. Han and D.Padua,"Parallel Mining of Closed Sequential Patterns," Proc. 11th ACM SIGKDD Int'l Conf. Knowl- edge Discovery and Data Mining (KDD '05), pp. 562-567, Aug. 2005.

[8] J. Wang and J. Han, "Bide: Efficient Mining of Frequent Closed Sequences," Proc. 20th Int'l Conf. Data Eng. (ICDE '04), pp. 79-91, 2004.

[9] X. Yan, J. Han, and R. Afshar, "Clospan: Mining Closed Sequential Patterns in Large Datasets," Proc. Third SIAM Int'l Conf. Data Mining (SDM '03), pp. 166-177, May 2003.

[10] M.N. Garofalakis, R.Rastogi, and K.Shim, "Spirit: Sequential Pattern Mining with Regular Expression Constraints," Proc. 25th Int'l Conf. Very Large Data Bases (VLDB '99), pp. 223-234, 1999.

[11] C. Luo and S.M. Chung, "Efficient Mining of Maximal Sequential Patterns Using Multiple Samples," Proc. Fifth SIAM Int'l Conf. Data Mining (SDM), 2005.

[12] H.Cao, N.Mamoulis, and D.W. Cheung, "Mining Frequent Spatio-Temporal Sequential Patterns," Proc. Fifth Int'l Conf. Data Mining (ICDM '05), pp. 82-89, Nov. 2005.

[13] J.-K. Guo, B.-J. Ruan, and Y.-Y. Zhu, "A Top-Down Algorithm for Web Log Sequential Pattern Mining," Proc. Ninth Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD), 2005.

[14] C.-C. Ho, H.-F. Li, F.-F. Kuo, and S.-Y. Lee, "Incremental Mining of Sequential Patterns over a Stream Sliding Window," Proc. IEEE Int'l Workshop Mining Evolving and Streaming Data (IWMESD '06), Dec. 2006