A Data Warehousing and Data Mining Project report on

# Mining weighted sequential patterns in a sequence database with a time-interval weight

*under the guidance of*

**Dr. M. Venkatesan**

*Submitted by*

**Tejas R**

**16CO148**

**Prajval M**

**16CO234**

**Soham Patil**

**16CO249**

**VII Sem B.Tech**

**Department of Computer Science and Engineering**

**National Institute of Technology Karnataka, Surathkal.**

***November 2019***

## Introduction

Sequential pattern mining aims to discover interesting sequential patterns in a sequence database, and it is one of the essential data mining tasks widely used in various application fields such as Web access pattern analysis, customer purchase pattern analysis, DNA sequence analysis, etc. In general, sequential pattern mining finds all frequent sequential patterns for a given sequence database and a frequency threshold whose occurrence frequency is no less than the threshold. In many of the previous researches on sequential pattern mining problems, sequential patterns and items in a sequential pattern have been considered uniformly. However, they have a different importance in real world applications, and thus more interesting sequential patterns can be found when the different importance is considered in sequential pattern mining. Based on this observation, weighted sequential pattern mining has recently been proposed and actively studied. In contrast to general sequential pattern mining based on simple support counting, the weight of information is used in finding interesting sequential patterns in weighted sequential pattern mining. For a sequence or a sequential pattern, not only the generation order of data elements but also their generation times and time-intervals are important.

Therefore, for sequential pattern mining,the time-interval information of data elements can help to get more valuable sequential patterns.Sequential pattern mining algorithms have been presented which consider a time-interval between two successive items in a sequential pattern. However, they simply consider a time-interval between two successive data elements as an item, and thus they are unable to get weighed sequential patterns considering different weights of sequences in a sequence database. If the importance of sequences in a sequence database is differentiated based on the time-intervals in the sequences, more interesting sequential patterns can be found.

## Background

Many studies have contributed to efficient mining of sequential patterns, such as general sequential pattern mining closed and maximal sequential pattern mining, constraint-based sequential pattern mining, approximate sequential pattern mining, sequential pattern mining in multiple data sources, sequence mining in biological data, incremental mining of sequential patterns, and sequential pattern mining in noisy data.

Among the algorithms for general sequential pattern mining,SPADE and PrefixSpan are more efficient than others in terms of processing time. SPADE is one of the vertical-format-based algorithms and uses equivalence classes in the mining process. PrefixSpan is one of the pattern-growth approaches. It recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only the locally frequent fragments.

To improve the usefulness of mining results in real world applications, weighted pattern mining has been studied in association rule mining and sequential pattern mining.Most of the weighted pattern mining algorithms usually require pre-assigned weights, and the weights are generally derived from the quantitative information and the importance of items in a real world application. For example, in a retail database, the quantity and price of the item being sold is considered as its weight. Time-interval and gap information between items in a sequence have been used to find interesting sequential patterns, and there have been several studies on mining sequential patterns considering them. Pei et al. and Ji et al. the two brothers have proposed constrained sequential pattern mining methods, which use time-interval and gap information as a constraint. In these methods, they are used only to confine the mining result of sequential patterns, but they do not give a mining result of weighted sequential patterns. Chen et al. have proposed sequential pattern mining algorithms, considering time-interval information between two successive items in a sequence, whose target database is a sequence database with single items and their corresponding time-intervals. However, the algorithms just consider time-interval information between two successive items as an item, so that they cannot support to get weighted sequential patterns based on different weights of sequences.

## Problem Definition

In generalized sequential pattern mining only the order of the data elements are considered. Whereas for mining sequential patterns using time-intervals , the generation order of the itemsets along with the timestamps of generation is considered.The time-intervals between the itemsets adds extra interestingness into the frequent patterns derived from the database.

The following is a mathematical definition of the problem. Let I= $\{i_1,i_2,...i_n\}$ be a set of items. A sequence S is defined as follows : S= $<s_1,s_2,...s_n>$ where $s_i$ is an itemset and S is an ordered list of itemsets. The sequence S is also associated with an ordered list of timestamps TS(S) =$<t_1,t_2,...t_n>$ where each $t_i$ corresponds to the timestamp of itemset $s_i$. A sequence database SDB is a set of tuples of the form <sid,S> where sid is the sequence identifier.The following is an example of a SDB :

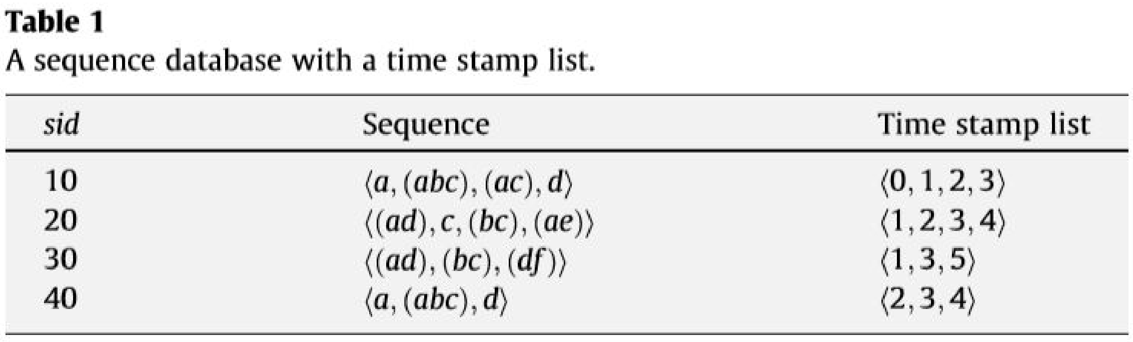


**Table 1**
A sequence database with a time stamp list.

| *sid* | Sequence | Time stamp list |
|---|---|---|
| 10 | $\langle a,(abc),(ac),d\rangle$ | $\langle 0,1,2,3\rangle$ |
| 20 | $\langle (ad),c,(bc),(ae)\rangle$ | $\langle 1,2,3,4\rangle$ |
| 30 | $\langle (ad),(bc),(df)\rangle$ | $\langle 1,3,5\rangle$ |
| 40 | $\langle a,(abc),d\rangle$ | $\langle 2,3,4\rangle$ |

## Proposed algorithm

The proposed algorithm calculates the Time-Interval weights for each sequence in the SDB and if the TiW is greater than or equal to the minimum support min_support, the sequence is classified as a frequent pattern. To obtain the TiW of each sequence the following algorithm is implemented. For each sequence $s_i$ of S do the following :

1. **Itemset-pairs generation :**
   Generate all the pairs of itemsets for the given sequence $s_i$. The number of pairs for a sequence is ${}^{n}C_2$ where n is the number of itemsets in the sequence.

   Hence, **Number of pairs = n(n-1)/2**.

2. **Time interval between itemset pairs :**
   For each of the above pairs of items ($s_i$ , $s_j$ ) calculate the time intervals $T_{ij}$ as follows :

   $$\mathbf{T_{ij} = t_j - t_i}$$

   where $1 <= i < j <= n$. This condition ensures that the time interval $T_{ij}$ is always positive.The range of values for $T_{ij}$ is [1, ∞] as there is no upper bound to the timestamp. Hence to normalize the values of the time intervals, we use specific weighting functions defined below.The following table shows the pairs of itemsets and their respective time intervals for the first sequence in the example SDB :

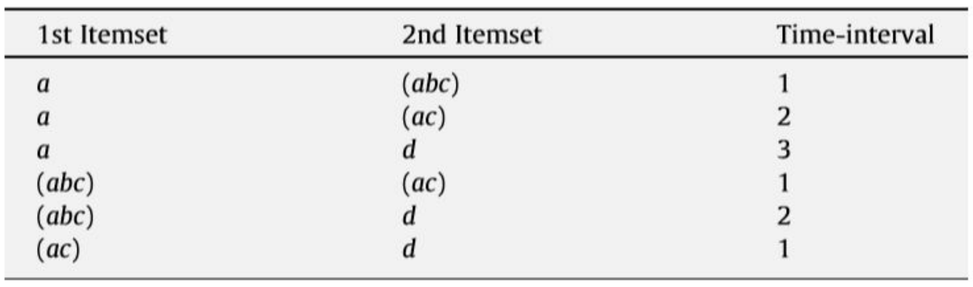


**Table 2**
Possible pairs of itemsets.

| 1st Itemset | 2nd Itemset | Time-interval |
|---|---|---|
| *a* | (*abc*) | 1 |
| *a* | (*ac*) | 2 |
| *a* | *d* | 3 |
| (*abc*) | (*ac*) | 1 |
| (*abc*) | *d* | 2 |
| (*ac*) | *d* | 1 |

3. **Time-interval weights of a itemset-pairs:**
   Since the time intervals don’t have an upper bound, weighting functions are used to normalize the time intervals.The following 3 weighting functions are proposed by the paper :

   (i) *General scale weighting:* $w_g(TI_{ij}) = \delta^{\frac{TI_{ij}}{u}} = \delta^{\frac{t_j - t_i}{u}}$ *[WF_1].*

   (ii) *Log scale weighting:* $w_l(TI_{ij}) = \delta^{\log_2\left(1+\frac{TI_{ij}}{u}\right)} = \delta^{\log_2\left(1+\frac{t_j - t_i}{u}\right)}$ *[WF_2].*

   (iii) *General scale weighting with a ceiling:* $w_c(TI_{ij}) = \delta^{\left\lceil\frac{TI_{ij}}{u}\right\rceil} = \delta^{\left\lceil\frac{t_j - t_i}{u}\right\rceil}$ *[WF_3].*

   Let u be the size of unit time and generally u>0 and delta be the base for time reduction.δ ranges from (0,1).Here u and δ are the constants used to change the behavior of the weighting functions. Smaller the values of u and δ , more sensitive the weighting functions to small variations in the time intervals. The following is the behaviour of the weighting functions :

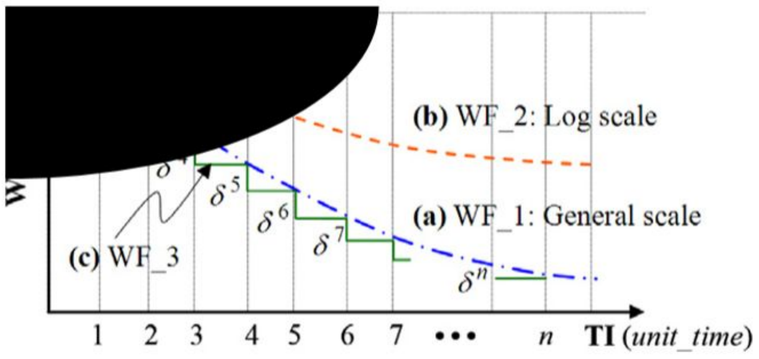


**Fig. 1.** Time-interval weighting functions.

4. **Strength of itemset-pairs:**
   The strength of itemset-pair $s_i$ and $s_j$ is defined as the product of the number of items in $s_i$ and $s_j$ .

   **$ST_{ij}$ = length($s_i$) x length($s_j$)**

   Where length($s_i$) denotes the number of items in $s_i$.The concept of strength is introduced for the following reason.Consider a scenario in which 2 itemset-pairs have the same TiW. Let the itemset-pairs be as follows :

   1. pair1= a,(ab)
   2. pair2= (abc),(abcd)

Now, even though the pairs have the same TiW, we should give more importance to pair2 as there is more association between items in pair2. Hence there arises the need to take into account the size of the itemsets .

5. **TiW of a sequence:**
The TiW of a sequence is calculated as the weighted mean of the TiWs of the different itemset pairs with $ST_{ij}$ used as weights.

$$W(S) = \begin{cases} \frac{1}{N}\sum_{i=1}^{l-1}\sum_{j=i+1}^{l}\{w(TI_{ij}) \times ST_{ij}\}, \text{ where } N = \sum_{i=1}^{l-1}\sum_{j=i+1}^{l} ST_{ij} & \\ \text{and } w(TI_{ij}) \text{ denotes a weighting function} & (l \geqslant 2) \\ 1 & (l = 1) \end{cases}$$

Where $w(TI_{ij})$ is the weight of the itemset-pairs and $ST_{ij}$ is the strength of itemset pairs. After calculating the TiW of a sequence , the next step in the algorithm is to calculate the support of a sequence.Here the concept of super-sequences come into play.

6. **Super sequence of a sequence :**
Consider 2 sequences $A=<a_1,a_2,...a_n>$ and $B=<b_1,b_2,...b_m>$ . B is said to be a super sequence of A if there exists integers $1<=j_1<j_2<...j_n<=m$ such that $a_i \subseteq b_{ji}$,
For all i $\epsilon$ [1,n].

7. **TiW support of a sequence :**
The TiW support of a sequence X is calculated by summing all the TiWs of the sequences S belonging to the SDB which are super-sequences of X and then dividing the sum by the total TiWs of the sequences.

$$TiW\text{-}Supp(X) = \frac{\sum_{S:(X \subseteq S)\wedge(S \in SDB)} W(S)}{\sum_{S:S \in SDB} W(S)}.$$

8. **TiW sequential patterns :**
The TiW sequential patterns are filtered using the threshold value defined by the min_support. Min_support is always 0<min_support<=1 . A sequence X is a time-interval weighted sequential pattern iff

**TiW_support(X) >= min_support**

Changing the value of min_support we can vary the number the frequent sequential patterns mined.

## Comparison between Simple support and TiW support

Simple supports are calculated using the super sequences of the sequence but there are no weights associated with any sequence .Hence simple support is reduced to a ratio of the number of super-sequences of a given sequence and the total number of sequences.TiW support on the other hand also takes into consideration the weights associated with the sequences and we know that these TiW weights are less than or equal to 1.Hence the TiW-support(X) <= Simple-support(X) .This can be observed in the following comparison table :

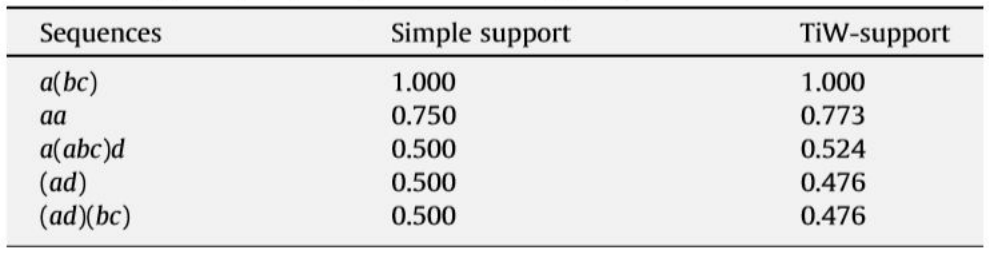


**Table 4**
Change of supports (Simple support vs. TiW-support).

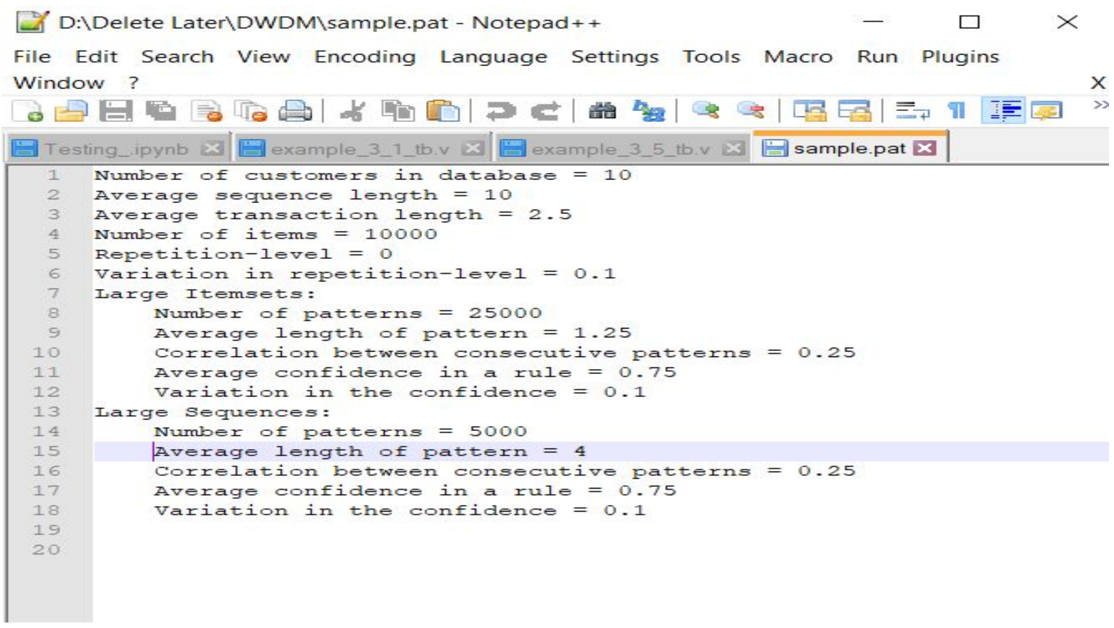
| Sequences | Simple support | TiW-support |
|---|---|---|
| $a(bc)$ | 1.000 | 1.000 |
| $aa$ | 0.750 | 0.773 |
| $a(abc)d$ | 0.500 | 0.524 |
| $(ad)$ | 0.500 | 0.476 |
| $(ad)(bc)$ | 0.500 | 0.476 |

# Dataset Generation

We are generating synthetic dataset from IBM for generating our dataset. The generator uses probability to distribute the data. The approach is to use a probability distribution function or a randomisation function, the approaches never affect the performance and this has been proven in the literature. This dataset almost never affects the performance of the new framework of TiWS pattern mining.

The parameters used for generating the patterns are number of customers, average sequence length, Number of Items, Average transaction length. Repetition level.
For generating large itemsets we generate high number of patterns and average number of patterns.For large sequences we increase the number of patterns and average length of patterns.

Following is a snapshot of the parameters of the sample dataset generated

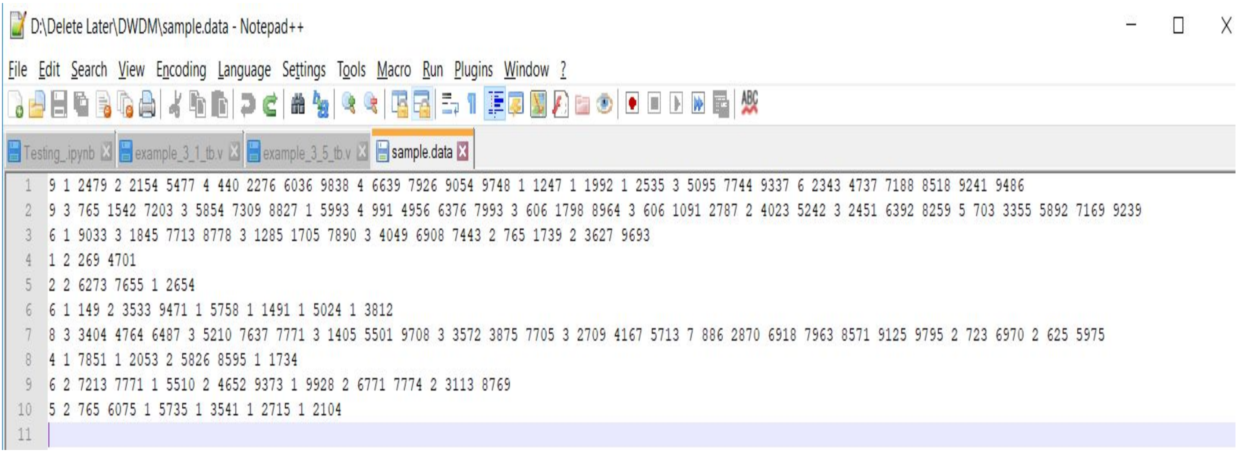

D:\Delete Later\DWDM\sample.pat - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Testing_.ipynb
example_3_1_tb.v
example_3_5_tb.v
sample.pat

```
Number of customers in database = 10
Average sequence length = 10
Average transaction length = 2.5
Number of items = 10000
Repetition-level = 0
Variation in repetition-level = 0.1
Large Itemsets:
    Number of patterns = 25000
    Average length of pattern = 1.25
    Correlation between consecutive patterns = 0.25
    Average confidence in a rule = 0.75
    Variation in the confidence = 0.1
Large Sequences:
    Number of patterns = 5000
    Average length of pattern = 4
    Correlation between consecutive patterns = 0.25
    Average confidence in a rule = 0.75
    Variation in the confidence = 0.1
```

An efficient storing of the data is given bellow:

- The line number in the file gives the transaction-id or the sequence id.
- The first digit of each line gives the number of itemsets present in the sequence.
- The digits following the number of itemsets is actually a pair, first digit (n) contains the number of items in the dataset, whereas the following n digits are the items in the dataset.

A snapshot of sample dataset generated is given below:

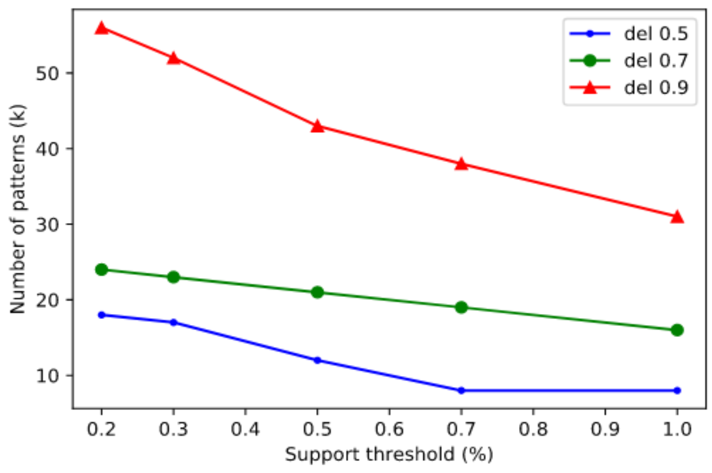

D:\Delete Later\DWDM\sample.data - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Testing_.ipynb
example_3_1_tb.v
example_3_5_tb.v
sample.data

```
9 1 2479 2 2154 5477 4 440 2276 6036 9838 4 6639 7926 9054 9748 1 1247 1 1992 1 2535 3 5095 7744 9337 6 2343 4737 7188 8518 9241 9486
9 3 765 1542 7203 3 5854 7309 8827 1 5993 4 991 4956 6376 7993 3 606 1798 8964 3 606 1091 2787 2 4023 5242 3 2451 6392 8259 5 703 3355 5892 7169 9239
6 1 9033 3 1845 7713 8778 3 1285 1705 7890 3 4049 6908 7443 2 765 1739 2 3627 9693
1 2 269 4701
2 2 6273 7655 1 2654
6 1 149 2 3533 9471 1 5758 1 1491 1 5024 1 3812
8 3 3404 4764 6487 3 5210 7637 7771 3 1405 5501 9708 3 3572 3875 7705 3 2709 4167 5713 7 886 2870 6918 7963 8571 9125 9795 2 723 6970 2 625 5975
4 1 7851 1 2053 2 5826 8595 1 1734
6 2 7213 7771 1 5510 2 4652 9373 1 9928 2 6771 7774 2 3113 8769
5 2 765 6075 1 5735 1 3541 1 2715 1 2104
```

## Experimental results

Various experiments were carried out on the generated dataset and a comparison between the different weighting functions,the effect of δ and u on the number of patterns were studied in detail. The IBM dataset generator tool was used to generate 100 sequences.The values of δ ,u and support threshold (%) were varied and the following were observed :

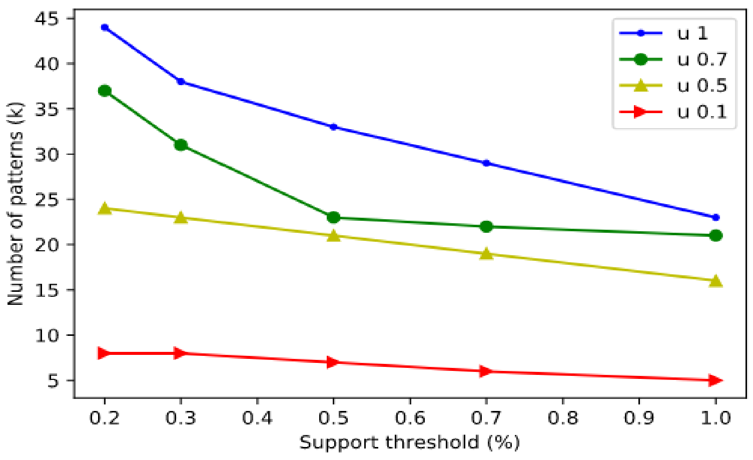
- Effect of δ on the number of patterns:



The following 2 observations can be made from the above graph:

1. The number of sequential patterns returned by the algorithm decreases as the value of δ decreases.This is because as δ decreases , the base part of the weighting function decreases and the weighting function becomes more sensitive to the $TI_{ij}$ and hence $w(TI_{ij})$ becomes low resulting in a low value of TiW-support.Due to this the number of sequences returned is low.
2. The number of patterns returned by the algorithm decreases as the support threshold increases.This is explainable as the higher the support threshold, lesser the number of sequences satisfying the condition.

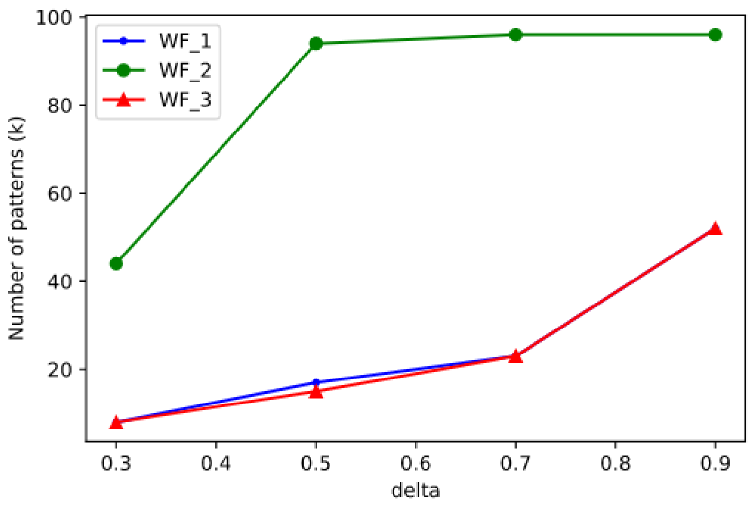
- Effect of u on the number of patterns:



The following 2 observations can be made from the above graph:

1. The number of sequential patterns returned by the algorithm decreases as the value of u decreases.This is because as u decreases ,the exponent part of the weighting function increases and the weighting function becomes more sensitive to the $TI_{ij}$ and hence $w(TI_{ij})$ becomes low resulting in a low value of TiW-support.Due to this the number of sequences returned is low.
2. The number of patterns returned by the algorithm decreases as the support threshold increases.This is explainable as the higher the support threshold, lesser the number of sequences satisfying the condition.

- Effect of the weighting function on the number of patterns:

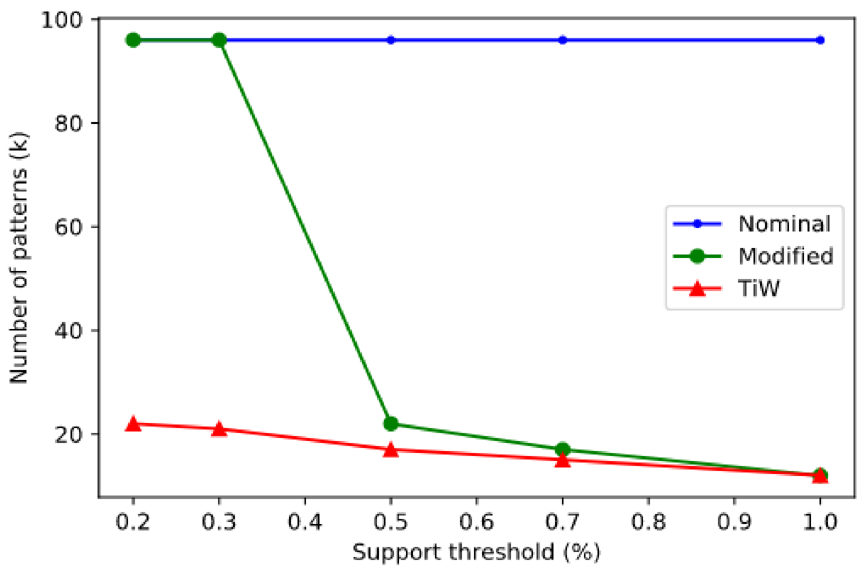


The following 2 observations can be made from the above graph:

1. The number of patterns returned by the algorithm increases as δ increases as established before.
2. The weighting function WF_2 returns larger number of sequential patterns than the weighting functions WF_1 and WF_3.This is because WF_2 uses logarithm in the exponent component due to which the exponent component reduces. As a result of this w($TI_{ij}$) increases resulting in an increase in the TiW-support of the sequence.Hence the number of sequential patterns returned increases.

## Our Contributions

1. Proposed a modified version of TiW support algorithm with a weightage $x$ given to TiW support and (1-$x$) given to simple support.



In some cases based upon the domain and patterns available Nominal Frequent Pattern mining will give more number of patterns where many of them might not be much significant and Time Weighted Average might give very few patterns which might become difficult for analysis. Hence a modified algorithm called Modified TiW algorithm is proposed by us which gives in a mixture of both of these. It can be seen in the graph above that the number of patterns recognised is always between Nominal and TiW. Hence having a parameter x can give us the desired number of patterns, x however has to be chosen by the domain expert.

**Modified TiW-support(X)= $x$*TiW-support(X) + (1-$x$)*simple-support(X)**

This ensures that the number of patterns returned by the algorithm is never too low thereby assuring better association rules between the itemsets.

2. We have also made the algorithm more scalable by keeping track of the total TiW supports of the SDB at any point in time. The above paper hasn't considered the fact that all the SDBs are real-time in the sense that , the data is always getting added to the SDB. The above paper hasn't addressed this problem , so the TiW supports have to be calculated every time a new sequence is added into the database to find the sequential patterns.

   We have considered the above problem and proposed a solution to tackle it and make the system more scalable. The following is the algorithm :

## Proposed algorithm

Let S be the SDB initially and S' be the SDB after addition of a new sequence. Let the new sequence added be s. Let sum_TiW(D) denote the sum of TiWs of all the sequences in the sequence database D.

1. Initially calculate the TiW supports of all the sequences using the above algorithm.Store the total TiW supports of all the sequences in a variable. Let sum_TiW(S) denote the sum.
2. On addition of a new sequence s into the database, calculate the TiW support for s. Let this be TiW(s).
3. Let is_super_sequence(a,b) be a function which returns true if a is a super sequence of b and false otherwise .For all the original sequences in S , modify the TiW supports using the following formula :

**TiW-support-new(X) = [TiW-support-old(X) * sum_TiW(S) + is_super_sequence(s,X) * TiW(s)] / sum_TiW(S')** _____ **(1)**

**sum_TiW(S') =sum_TiW(S) + TiW(s)** _____ **(2)**

Using the proposed algorithm we save one scan on the entire database which would have otherwise been used for the calculation of TiW-support-new(X).

## Conclusion

In TiW each sequence itemsets have corresponding timestamps which is used to calculate the weight of the sequence and in turn the TiW support of the sequence.
In the conventional support (nominal support) algorithm for getting frequent patterns only the item ordering in a sequence are considered whereas on TiW the time-intervals are also taken into account and thus helps in generating better frequent patterns.

In TiW algorithm time unit and delta are two parameters used. Both the parameters are less than 1. There algorithms can be used in calculating the weights of the sequence simple scale, log scale or taking the ceiling. Log scale provides more frequent patterns than simple scale.Using TiW gives interesting and more valuable sequential patterns. In the real world domains , not only the generation order but also the generation times and the generation intervals play a significant role.

We have also proposed another algorithm called modified TiW algorithm which can be used in any cases and fine tuned with a parameter x to get the number of frequent patterns necessary for the analysis. Further the system is made scalable using the proposed algorithm which improves the computation time of updating the TiW-supports of the sequences. Further improvements for this paper can be done. Currently we don't have an algorithm to figure out the values of delta and time unit (u) for the TiW algorithm and x for modified TiW algorithm and a domain expert is required to select these values. On careful analysis of the trends ,a specific value of u , δ and x can be chosen which would give the most optimal number of sequential patterns.

## References

[1] R. Agrawal, R. Srikant, Mining sequential patterns, in: Proceedings of the 1995 International Conference on Data Engineering (ICDE ’95), 1995, pp. 3–14.
[2] C.H. Cai, A.W.C. Fu, C.H. Cheng, W.W. Kwong, Mining association rules with weighted items, in: Proceedings of the IEEE International Database Engineering and Applications Symposium (IDEAS ’98), 1998, pp. 68–77.
[3] E. Chen, H. Cao, Q. Li, T. Qian, Efficient strategies for tough aggregate constraint-based sequential pattern mining, Information Sciences 178 (6) (2008) 1498–1518.
[4] Y.-L. Chen, M.-C. Chiang, M.-T. Ko, Discovering fuzzy time-interval sequential

patterns in sequence databases, IEEE Transactions on Systems Man and
Cybernetics – Part B: Cybernetics 35 (5) (2005) 959–972.
[5] Y.-L. Chen, T.C.-H. Huang, Discovering time-interval sequential patterns in
sequence databases, Expert Systems with Applications 25 (1) (2003)
343–354.
[6] H. Cheng, X. Yan, J. Han, IncSpan: incremental mining of sequential patterns in
large databases, in: Proceedings of the 2004 ACM SIGKDD International
Conference on Knowledge Discovery and Data Mining (KDD ’04), 2004, pp.
527–532.
[7] M. Ester, A top-down method for mining most specific frequent patterns in
biological sequence data, in: Proceedings of the 2004 SIAM International
Conference on Data Mining (SDM ’04), 2004, pp. 90–101.
[8] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.-C. Hsu, FreeSpan: frequent
pattern-projected sequential pattern mining, in: Proceedings of the 2000 ACM
SIGKDD International Conference on Knowledge Discovery and Data Mining
(KDD ’00), 2000, pp. 355–359.
[9] X. Ji, J. Bailey, G. Dong, Mining minimal distinguishing subsequence patterns
with gap constraints, Knowledge and Information Systems 11 (3) (2007) 259–
296.
[10] H.-C. Kum, J. Pei, W. Wang, D. Duncan, ApproxMAP: approximate mining of
consensus sequential patterns, in: Proceedings of the 2003 SIAM International
Conference on Data Mining (SDM ’03), 2003, pp. 311–315.
[11] H.-C. Kum, J.H. Chang, W. Wang, Sequential pattern mining in multi-databases
via multiple alignment, Data Mining and Knowledge Discovery 12 (2+3)
(2006) 151–180.
[12] M.-Y. Lin, S.-C. Hsueh, C.-W. Chang, Fast discovery of sequential patterns in
large databases using effective time-indexing, Information Sciences 178 (22)
(2008) 4228–4245.
[13] S. Lo, Binary prediction based on weighted sequential mining method, in:
Proceedings of the 2005 International Conference on Web Intelligence, 2005,
pp. 755–761.
[14] C. Luo, S.M. Chung, Efficient mining of maximal sequential patterns using
multiple samples, in: Proceedings of the 2005 SIAM International Conference
on Data Mining (SDM ’05), 2005, pp. 64–72.
[15] J. Pei, J. Han, W. Wang, Mining sequential patterns with constraints in large
databases, in: Proceedings of the 2002 ACM International Conference on
Information and Knowledge Management (CIKM ’02), 2002, pp. 18–25.
[16] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu,
Mining sequential patterns by pattern-growth: the PrefixSpan approach, IEEE
Transactions on Knowledge and Data Engineering 16 (11) (2004) 1424–1440.
[17] K. Sun, F. Bai, Mining weighted association rules without preassigned weights,
IEEE Transactions on Knowledge and Data Engineering 20 (4) (2008) 489–495.
[18] F. Tao, F. Murtagh, M. Farid, Weighted association rule mining using weighted
support and significance framework, in: Proceedings of ACM SIGKDD ’03,
2003, pp. 661–666.
[19] P. Tzvetkov, X. Yan, J. Han, TSP: mining Top-K closed sequential patterns,

Knowledge and Information Systems 7 (4) (2005) 438–457.
[20] J. Wang, J. Han, C. Li, Frequent closed sequence mining without candidate maintenance, IEEE Transactions on Knowledge and Data Engineering 19 (8) (2007) 1042–1056.
[21] K. Wang, Y. Xu, J.X. Yu, Scalable sequential pattern mining for biological sequences, in: Proceedings of the 2004 ACM International Conference on Information and Knowledge Management (CIKM '04), 2004, pp. 178–187.
[22] W. Wang, J. Yang, P.S. Yu, WAR: weighted association rules for item intensities, Knowledge and Information Systems 6 (2) (2004) 203–229.
[23] X. Yan, J. Han, R. Afshar, CloSpan: mining closed sequential patterns in large datasets, in: Proceedings of the 2003 SIAM International Conference on Data Mining (SDM '03), 2003, pp. 166–177.
[24] J. Yang, P.S. Yu, W. Wang, J. Han, Mining long sequential patterns in a noisy environment, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02), 2002, pp. 406–417.
[25] U. Yun, A new framework for detecting weighted sequential patterns in large sequence databases, Knowledge-Based Systems 21 (2) (2008) 110–122.
[26] U. Yun, An efficient mining of weighted frequent patterns with length decreasing support constraints, Knowledge-Based Systems 21 (8) (2008) 741–752.
[27] M.J. Zaki, SPADE: an efficient algorithm for mining frequent sequences, Machine Learning 42 (1/2) (2001) 31–60.