*I.J. Education and Management Engineering,* **2015, 4, 13-20**
Published Online September 2015 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2015.04.02

***Available online at http://www.mecs-press.net/ijeme***

# Shift Window FPTree - An Efficient Stream Mining Algorithm

Deepak K Mishra[a], Dr. Varsha Sharma[b]

[a]*PG Scholar, School of Information Technology, RGPV, Bhopal MP-462036, India*
[b]*Professor, School of Information Technology, RGPV, Bhopal MP-462036, India*

**Abstract**

Breathless flow in data collection and storage mechanism has enabled Firms to heap up a massive amount of data. In many cases, these huge volumes of data can be mined for fascinating and applicable information in a wide range of applications. When the arrival of data is fast as well in a large bunches in term of amount, this lead major problem to go through this data in both the circumstances in store it and in extracting the useful information from it. To taking under these issues continues mining or stream mining is a best way. Data steam mining allows to not storing the entire data for future prediction which lead to overcome the e-vestige and unnecessary storage overhead. But there is no such way in literature to mine continues data direct, so first one make it feasible accordingly and then mine it. Here in this paper we present an algorithm which handle stream data in very effective manner.

**Index Terms:** Data stream mining, flow of data, continues mining.



## 1. Introduction

Rapid advances in data collection and storage technology have enabled organizations to accumulate vast amount of data. Simple transactions of everyday life such as using a cash card, credit card, a telephone monitoring system or browsing the web lead to automated data storage. Currently, a wide class of data-intensive applications, in which data is uses in the form of continuous flow of streams, has been widely recognized. The basic challenge is that '*data-intensive*' mining is constrained by bounded availability of resources, time, memory, and data sample size. The size of data that has arrived and will approach in the future is extremely large; in fact, the series of arrival is potentially infinite. Thus, it is almost impossible to store it all, which allows researcher for introducing some techniques to deal with this kind of irrelevant data storage, due to these researches focused to extract important knowledge over continues data and only store important part of data. There are many more work done in this field but most of them having the limitation of time domain

* Corresponding author
Email: dkm05mishra@gmail.com, varshasharma@rgtu.ac.in

windows. Here in this paper we present an algorithm for extracting important dataset from continues data stream. The contribution of this paper is in presenting the Shift Window Frequent Pattern Tree (SW-FPTree) algorithm, which provides an efficient solution to the search for frequent itemsets in a stream of transactions when a time horizon is given.

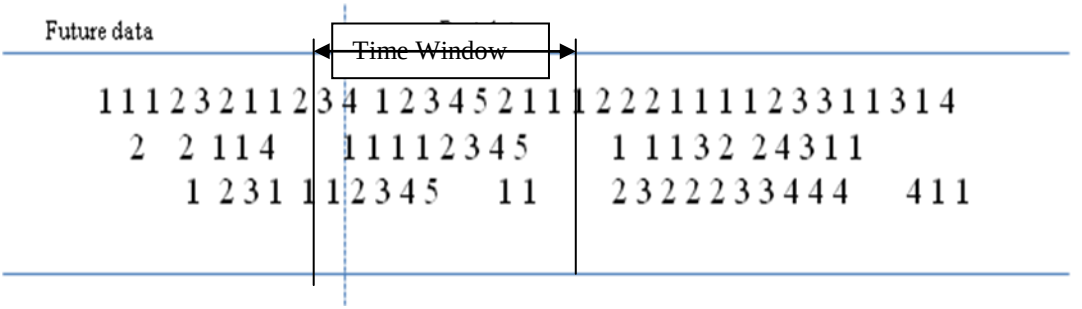


Fig.1 A Stream of Transactions Flowing Across a Time Window.

This paper broadly organized in the bellow manner: - **Part 2** Introduction of some basic methods and properties, **Part 3** Here we define the brief overview of related work and research, **Part 4** Description of the architecture of the algorithm, **Part 5** presentation of result of algorithm over some datasets and comparison with similar algorithms, **Part 6** Conclusion and future work.

## 2. Preliminary

### *2.1. Data Stream Mining*

Recently, database and data mining communities have cantered on a replacement knowledge model, wherever knowledge arrives within the type of continuous streams. It's typically spoken data streams or streaming knowledge. Mathematically data stream [2] defined as:

It is a unconditional sequence of transactions that is S= {*T*1, *T*2, *T*3, …, *T*N} where $T_I$ =<TID, e> with TID is transaction Id and e is set of items and N is current transaction number.

### *2.2. Sliding Window*

The key part in mining data stream is to change their property so they behave like static datasets and to face these constraints, specific attention has been paid to the efficient mining of frequent item-sets in data streams. According to [6], we can group the approaches into three main categories: landmark-, damped- and sliding-window based mining. A sliding window is defined as a fixed or variable group of item sets SW= {$I_1$, $I_2$, $I_3$, …, $I_H$} here I =<TID, e> and H is length of window in other word we also say that, a transaction-sensitive sliding window *SW* = [*T*N_H+1, *T*N_H+2, …, *T*N] is a window that slides forward for every incoming transaction, where N is TID of latest incoming transaction and H is the size of *SW*.

### *2.3. FP-Tree*

The FP-tree provides a base structure to facilitate mining in a static batch environment. The frequency of an item-set I over a time period T is the number of transactions in T in which I occurs. The support of I is the frequency divided by the total number of transactions observed in T. Let the minimum support be σ and the *relaxation ratio* be ρ=Ɛ/σ, where Ɛ is the maximum support error. I is frequent if its support is no less than σ; it

is sub-frequent if its support is less than σ but no less than Ɛ; otherwise, it is infrequent.

## 3. Related Work

### *3.1. Frequent Itemset Mining*

The mining of frequent pattern traditionally is being associated with association rule mining, as they provide the frequent pattern of items and this can be divided into two parts:-

1. To find the frequent items.
2. To define the association rules over frequent datasets.

There are two most commonly used and famous algorithms used for mining frequent items over static databases namely: *Apriori* [1] and *FP-Growth* [4]. In these two FP-Growths in more efficient in term of time and memory over Apriori, Although Apriori is the first and most commonly known algorithm. The main idea of it is to build a new type of data structure of frequent items called FP-Tree which is memory efficient.

### *3.2. Window Itemset Shift (WIS)*

Here is associate algorithm supported Apriori for knowledge stream mining referred to as Apriori with Window check and Window Itemset Shift(WIS) [7] that is given in algorithm 1, supported the window check, as advised by Propositions of windowing. Here in this case, the mining of frequent knowledge sets is additionally explained by both conditions- $\exists TI \in WS: I \subseteq TI$ and (ii) $supp(I) \geq S$ (see algorithm). as a result of the support condition entails the prevalence condition, it might seem to be perennial to think about each. However, testing the prevalence condition initial could be a quicker way to strain non-frequent data sets, which proves to be a crosscut to the conjunction. This characteristic may be exploited to create a faster on-line mining algorithm. At the start of the algorithm, the best window set is found, and also the lists candidate(C) and frequent pattern (F) area unit initialized by considering the records that area unit initially in Window. List C might be initially obtained by Algorithm1 by considering solely the prevalence condition, i.e., $C_{K+1} = I \in G_{K+1}$, therefore reposeful the condition that's expressed in algorithm1. Indeed, the prevalence condition is important however isn't adequate to elucidate the frequent data sets. However, this approach appearance appealing in theory, it's not possible in several circumstances of practical interest. The sole prevalence condition makes C explode within the variety of candidates, which reaches, in some cases, the entire set of data sets that consists of 2n components, where n is the variety of different items. A higher possibility is to think about frequent data sets because the initial candidates for the progress. The online learning method that's utilized by WIS can embody candidates that aren't thought-about at the start. Algorithms that area unit devoted to mine frequent itemsets expeditiously, such as FP-Growth [11] (applied to transactions among the sliding window), will improve the algorithm start-up. However, this issue is outside of the main target of this work, and cannot be thought-about more here.

**Algorithm 1** Window Itemset Shift (WIS)

1: C: candidate list
2: F: frequent itemset list
3: S: support threshold
4: W: window size
5:
6: find optional $W_S$
7: F← frequents (W, S)

8: C← candidates (W, $W_S$)
9: for all I Ɛ C do
10: build vector $v_I$, last(I) ← leftmost occurrence in $W_S$
11: end for
12: loop
13: move W forward of h slots
14: for all I Ɛ C do
15: last (I) +=h
16: if last (I) > length ($W_S$)
17: remove I from C
18: end if
19: end for
20: $R_S$ ←records entering $W_S$
21: for all Ir, Ir !=ϕ, Ir ⊆ r r Ɛ R do
22: if Ir Ɛ C then
23: last (Ir) = position (r)
24: else
25: insert Ir in C
26: build vector $v_I$, last (Ir) ← leftmost occurrence in $W_S$
27: end if
28: end for
29: F = {I Ɛ C | supp (I) >= S}
30: end loop

In addition to considering a sharp horizon, we consider a smooth window. Window Itemset Shift (WIS) as an alternative solution, which retains a memory of flowing candidates within a reduced test window as compression with past methods it is more advantageous, although it is more demanding in terms of the memory occupation. This relationship arises because of the additional data structures, which avoid re-exploring the entire item-set lattice. Indeed, in many applications that are of practical interest, not all of the time slots have the same relevance, e.g., more recent slots can be more interesting than older slots. Smoothness can be determined in both qualitative and quantitative terms. The most immediate approach is to assign a degree of relevance to each slot on a subjective basis. This approach will lead to shape a trapezoidal window. Other approaches could be related to the structural characteristics of the stream.

## 4. Proposed Algorithm

There are many methods/ algorithm available which deals with continues transactional data. But most of those methods use the basics of Apriori algorithm with many other parameters. The major problem with Apriori is access of databases many more times for finding the frequent pattern or frequent item sets. Due to the multiple access of database Ariori is suffer with memory efficiency and hence time efficiency problems. The FP-Growth [4] algorithm is another mostly used algorithm with the static datasets, and FP-Growth uses a new type of data structure for storing the frequent datasets named FP-Tree. The good think about this is it uses its local datasets for finding the frequent pattern thus this only allows one access of database which reduces the multiple memory access problem but there is a problem with this as well, that is to organizing the whole tree in main memory, latter this problem overcomes by use of partial structure and there is one more way to deal with this to store only the frequent data in tree rest will be pruned out. Thus the FP-Growth algorithm is more efficient in term of both the major parameters over static data set. There is some algorithms also available in literature which are based on the concept of FP-Growth over continues datasets like FP-Stream [3]. But they work with the time stamp in other words they handle the datasets over time frame which have a major issue of

the overload of the datasets, as the rate of data arrival is not fixed in most of the cases, which lead the huge amount of transaction in any particular frame or very less amount of transaction in some frames both the situations are not good as one lead to slower the performance and other may used unnecessary resources. So the based way to handle stream data is go through the frequency of transaction. The proposed algorithm Shift Window FPTree (*SW FPT*) is based on the *transaction sensitive* sliding window approach followed by the key of the FP-Growth. The SW FP-Tree is a frequency based data stream mining algorithm, which have the primary feature of the FP-Tree used as the main data structure and this uses the minimum support only feature of the frequent pattern mining and there is no overhead of the candidate generation and confidence count. Steps of the SW FPT are given bellow in algorithm2:

**Algorithm 2:** Shift Window FPTree (SW FPT)

1. Window W
2. Minimum support S
3. Transaction T
4. Item I
5. Window size h
6. 
7. Create an empty tree
8. Choose optimal window size $W_S$
9. Loop
10. Move W by h
11. For each transaction in W do
12. Built vector $V_I \forall I \varepsilon T$
13. Create a new node labelled $N_I$
14. if $N_I$ exist in tree then
15. $N_I \rightarrow$freq := +1
16. Else
17. Create a new link from root
18. End if
19. For each I $\varepsilon$ (W, $W_S$)
20. Remove I from Ws and insert into tree node
21. For each node $N_I$
22. If $N_I \rightarrow$freq $\geq$ S then
23. Insert into FPtree
24. Else
25. Prune out this node $N_I$
26. End if
27. End for
28. End for
29. Until no more transaction left
30. End

## 5. Result Analysis

### *5.1. Experiment setup*

The proposed algorithm and the previous related algorithm WIS is implemented and tested in following environment: - 64 bit system, OS- Windows7 Home Basic, processer- AMD A4-3330MX APU with 2.20 GHz

speed, RAM- 4GB, language used for implementation-java(jdk1.8.0_05), IDE- NetBeans8.0. The below two types of data sets used for testing the performance under the various circumstances: IBM's T10I4D100K [12] and c73d10k [13]. Experiment results of frequent pattern mining are shown in below charts in various conditions. Y-axis indicates the execution time (in ms).

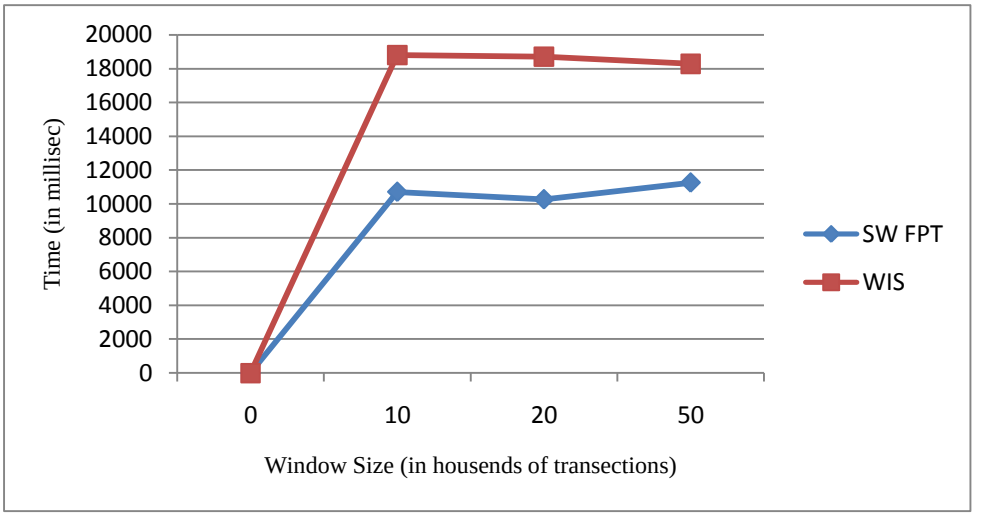


Fig.2a. Graph with Minimum Support 50% for Dataset T10I4D100K.

The figure 2-a, represents the output of algorithms for dataset T10I4D100K with minimum support 50%. X-axis refers the size of window and Y-axis refers the execution time in msec.

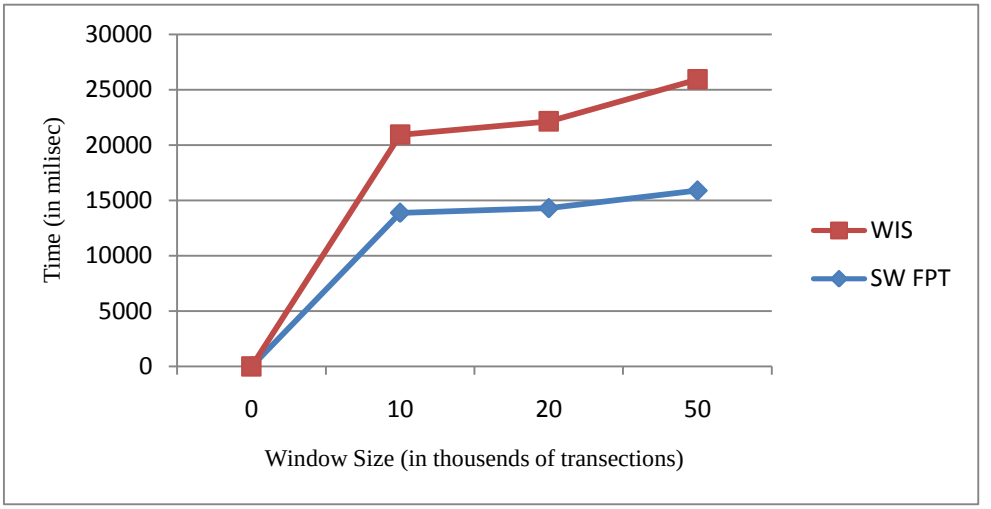


Fig.2b. Graph with Minimum Support 20% for Dataset T10I4D100K.

The figure 2b represents the output of algorithms for dataset T10I4D100K with minimum support 20%. X-axis refers the size of window and Y-axis refers the execution time in msec.

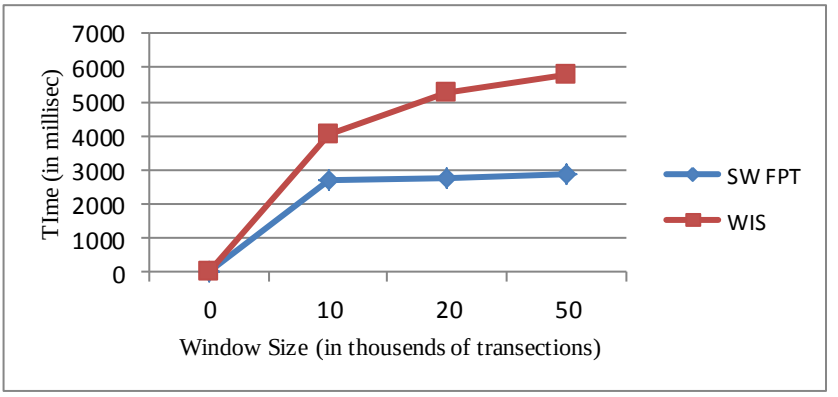


Fig.2c. Graph with Minimum Support 50% for Dataset c73d10k.

The figure 2-c represents the output of algorithms for dataset c73d10k with minimum support 50%. X-axis refers the size of window and Y-axis refers the execution time in msec.

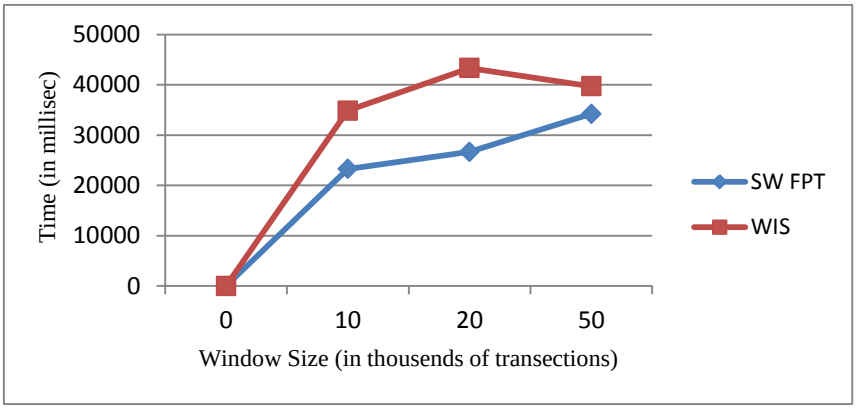


Fig.2d. Graph with Minimum Support 20% for Dataset c73d10k.

The figure 2-d represents the output of algorithms for dataset c73d10k with minimum support 20%.

The above comparative analysis charts are prepared in the following conditions with the execution time as Y-axis (in mille-seconds) and the sliding window as X-axis (window size in thousands no of transactions).

## 6. Conclusion

Here in this paper we present an algorithm for mining frequent pattern over continues data series. SW FPT is a robust and more efficient algorithm than its counter previous algorithms, it work smoothly over any kind of stream as it is frequency dependent instead of the time on the same time the algorithm is efficient enough to work with it. As the results explaining that the SW FPT is sufficient faster than WIS. Here we use WIS only in the counter because this is resent work in this field. The SW FPT is good enough to work with it, but then also there is a hole of some complex data structure and decision statements uses which can let the management of stack and memory overhead.

## References


[1] R. Agrawal, T. Imeilinski, A. Swami, "Database mining: a performance perspective", IEEE Trans. Knowl Data Eng. 5 (6) 914–925(1993).

[2] Charu C. Aggarwal "Data Streams: Models and algorithms", Springer 2007.

[3] Giannella. C, Han. J, Pei. J, et al., "Mining frequent patterns in data streams at multiple time granularities", In: Next Generation Data Mining. Cambridge, Massachusetts: MIT Press, pp. 11-212, 2003.

[4] Han, J.; Pei, J.; and Yin, Y, "Mining frequent patterns without candidate generation", ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00), 1.12 2000.

[5] J.H. Chang, W.S. Lee, "A sliding window method for finding recently frequent itemsets over online data streams", J. Inf. Sci. Eng. 20 pp. 753–762(2004).

[6] H.-F. Li, S.Y. Lee, mining frequent itemsets over data streams using efficient window sliding techniques, Expert Syst. Appl. 36 pp.-1466–1477 (2009).

[7] Luigi Troiano, Giacomo Scibelli, "Mining frequent itemsets in data streams within a time horizon". Data & Knowledge Engineering 89, pp.21–37, (2014).

[8] K Kiruba, Dr B Rosiline Jeetha, "A Comparative Study on Hierarchical Clustering in Data Mining", International Journal of Engineering Sciences & Research Technology 3(2), pp.656-659, feb 2014.

[9] Li Tu, Ling Chen "Finding Frequent Items over Data Stream". Journal of Computational Information Systems, 6 (12): 4127- 4134, 2010.

[10] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, Philip S. Yu "FP-stream Mining frequent patterns in data streams at multiple time granularities". 2002.

[11] Jiawei Han, Jian Pei, Yiwen Yin and Runying Mao, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach", Data Mining and Knowledge Discovery, 8, 53–87, 2004.

[12] http://www.almaden.ibm.com/software/quest/Resources.

[13] http://www.census.gov/.


## Author(s) Profile

**Deepak k Mishra** (born July 05, 1989) had been completed his master's degree (M. Tech.) with honors in Computer Science and Technology in May 2015 from School Of Information Technology, RGPV Bhopal India, and successfully completed his research work in the field of data stream mining, during this process he also published a study paper on various stream mining techniques.

**Dr. Varsha Sharma,** Varsha Sharma is working as Assistant Professor at RGPV Bhopal, India. She has 10 years teaching experience and has published 27 research papers in various International journals and conferences. Her research interests include wireless networks, data mining and cloud computing.

**How to cite this paper:** Deepak K Mishra, Varsha Sharma,"Shift Window FPTree - An Efficient Stream Mining Algorithm", IJEME, vol.5, no.4, pp.13-20, 2015.DOI: 10.5815/ijeme.2015.04.02