International Journal of Computer Science and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 4, Issue 4, Jun - Jul 2015, 23-34

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

IMPROVE FREQUENT PATTEREN MINING IN DATA STREAM

HIMANSHU M. SHAH[1] & NAVNEETKAUR[2]

[1]Master of Technology Pursing, Department of Computer Science, Lovely Professional University, Punjab, India

[2]Assistant Professor, Department of Computer Science, Lovely Professional University, Punjab, India

ABSTRACT

Frequent pattern mining is emerging and most interesting fields in Data mining. Application of Finding frequent item set is very wide like sensor network, web click stream data, and intrusion detection. A data stream is continuous, rapid, unbounded sequence of data. Mining Frequent pattern in stream data is very challenging because data can be scan one time only. Due to this reason traditional approach cannot be use for data stream. Frequent pattern mining generates enormous amount of frequent pattern. However producing all frequent pattern is not suitable. Finding only top- k pattern is more attractive whose utility is above a threshold. In addition considering weight factor with support is more realistic approach. Our algorithm finds can efficiently find potential top k high utility pattern and encompasses effective pruning mechanism. Our experiment results report that it outperforms previous algorithm in terms of runtime, memory usage.

KEYWORDS: Data Stream, High Utility Pattern Mining, Sliding Window, Frequent Itemset, Top K Pattern Mining

INTRODUCTION

Data mining is a technique to extract hidden useful information from large database. Here Useful pattern is frequent pattern. Item sets which satisfy minimum support threshold value is called Frequent pattern. Frequent pattern is very useful because it shows usefulness of item sets. There are many algorithms such as Apriori, FP Growth and éclat which can efficiently discover frequent pattern and trends from Database.

Now a days, Many Organization generates Hugh amount of data and very high speed in nature such as social website, sensor network and many other sources. Data which are rapid, unbounded and continuous in nature called as data stream. Data stream mining is very challenging and very hot field in data mining community.

Traditional mining technique is best for static database. Mining frequent pattern from dynamic database is very challenging and harder than static database. Essential Requirement for data stream are 1] Data can be scan ones. 2] Data arrives continuously so processing should be as fast as possible. 3] Data stream are very large, it keep on arriving but memory is limited so algorithm should uses limited or constant memory usage.4]Response time of user query should be minimal. Mining in data stream can generates hugh number of frequent pattern but the problem is limitation of storage and processing capability. Therefore instead of generating all frequent patterns, generate closed frequent pattern or Maximal Frequent pattern is very useful because it represent frequent pattern in more compact form. an item set is closed if there is no superset with same support. An item set is maximal if there is no superset.

In real world, each item has their importance like price or weight. Some item set which is less frequent but can generate higher profit. So item set with high profit is important for business. This is known as utility mining.

There are three data stream processing models which are Landmark, Damped and Sliding windows. Landmark

model mines all frequent item sets over the entire history of stream data from a specific time point called landmark to the present. The Damped model mines frequent pattern but in this model each transaction has weight which decrease with time therefore it is also known as time fading model. The sliding window model mines frequent item set over stream data by temporary storing part of data and process after that it removed and moves ahead for new data. In general, Sliding window model is used when recent information is more important than historical.

This paper will focus on the following sections. In Section 2 we discuss relevant research about this topic. In Section 3, preliminaries and definition are described. Section 4 Introduces Proposed Framework ICFP. In Section 5, the performance of ICFP is presented. At the end Conclusion and future work of this paper are discussed in section 6.

RELATED WORK

Frequent pattern mining is first start with static database. In Traditional algorithm, Aprior (Agrawal & Srikant,1994) and FP-growth (Han et. al.,2004) are pioneer algorithms. Apriori uses Breadth first search (BFS) and generate numbers candidate pattern in mining process and it need to perform scanning repeatedly. FP growth algorithm is based on Depth First Search (DFS) which improves mining process. FP-growth algorithm overcomes the problem of repeated scan and able to mine pattern with two fixed database scan. Till now, Most of algorithms are based on FP-growth.

These traditional algorithms are not suitable for data stream environment. Though FP-growth able to mine pattern with two scan, it is not suitable for data stream, Since data in data stream can be scan one time only .Based on this fact, There are three processing model proposed .

Sliding Window Based Frequent Pattern Mining Over Data Stream

Sliding window consider recent data as more important and based on that there are many approaches (ahmed et al., 2009; Chen et al., 2012; Deypir et al., 2012; Farzanyar et al., 2012; Li, 2011; Mozafari et al., 2008; Shie et al.,2012; Tanbeer et al., 2009b; Zhang & Zhang, 2011) have been proposed. An Efficient approach proposed by (Tanbeer et al., 2009a, 2009b),which uses tree restructuring technique, BSM which perform restructuring operation more effectively by path adjusting method, etc.. Algorithmest Win Proposed by Chang and Lee(2005) which finds recent frequent pattern and uses reduced minimum support threshold named significance to early monitoring of itemsets before they become actually frequent itemset. The Moment algorithm (Chi et al., 2006) finds closed frequent pattern bt maintaining a boundary between frequent closed itemset and other itemset.

Concept Drift

Koh and Lin(2009) proposed an innovative approach which continuously monitors the incoming transactions to detect the occurrence of a concept shift. The frequent itemset are mined when a concept shift observed. F.Nort et al.(2013) proposed an approache based on concept drift named TMoment which uses sliding window

Weighted Condition is Frequent Patter Mining Over Data Stream

In real world, Each item have their importance known as weight or utility(Price or profit).Not only support but weight also play crucial factor in mining process. But the main challenge is to maintain anti-monotone property. Due to the fact that weighted infrequent pattern can become weighted frequent which normally destroy that property, Many researcher able to maintain this property by variety of methods(Ahmed et al.,2009,2012;Wang & Zeng,2011;Yun & Ryu,2011;Yun et. al.,2011,2012).Chang and Lee(2006) introduced an algorithm called estDec based on time decay model in which each

transaction has a weight decreasing with age. In this method, in order to reduce the effect of old transactions in the set of frequent patterns a decay rate is defined. Up-Growth (Tseng et al., 2010) which is novel algorithm that uses various pruning and counting strategies during mining process. IWFP Algorithm (ahmed et al., 2012) is a weighted frequent pattern mining, Applying BSM method. THUI-Mine (Tseng et al., 2006) is the first algorithm for mining high utility itemsets. WFPMDS (Ahmed et al., 2009) mines weighted frequent pattern using sliding window. The algorithm produce recent mining result by using sliding window and uses tree restructuring work with BSM technique.

In this study, the framework of the proposed algorithm, IFP is based on T-HUDs mining algorithm

PRELIMINARIES

A data stream $D=\{B_1,B_2,\ldots,B_N\}$ is a an infinite sequence of batches where each batch Bi contains a set of transactions i.e. $Bi=\{T_1,T_2,\ldots ,T_K\}$ where k > 0.Each transaction $T=\{(i_1,q_1), (i_2,q_2), \ldots,(i_n,q_n)\}$ is a set of items where i represent an item such as $i \in I$ and q represent quantity of item in transaction. An itemset is a non-empty set of items. An itemset with size k is called kitemset. A window W is a sliding window which has number of batches $W = \{B1, B2, \ldots,B_m\}$.

Defination 1. Utility of an Item I in transactionTj

Tj is defined as :u(i,Tj)=q(i,Tj) x p(i) where q(I,Tj) is quantity of an item and p(i) is external utility of item

Definition 2. Utility of an itemset X in a transaction

Tj is defined by: $u(X,Tj) = \sum_{i\in X} u(I,Tj)$.

For example, $u(\{bc\},T_3)=2 \times 6 +3 \times 5 =27$ in fig 1.

Definition 3. Utility of an itemset X in a data set D

$u_D(X)= \sum_{X\subseteq Tj \wedge} Tj_{\in D} \sum_{i\in x} u(I,Tj)$.

We use u(X) to denote u_D(X0 when data set D is clear in the context.

Definition 4. Utility of a transaction

$T_j$is denoted as $TU(T_j)$ and computed as $u(T_j, T_j)$.

Definition 5. (High Utility Itemset (HUI))

An itemset X is called a high utility itemset (HUI) on a data set D if and only if $u_D(X) \geq$ min_util is called a minimum utility threshold.

Definition 6. Transaction-Weighted Utility (TWU)

TWU of an itmset X over a dataset D is defined as $TWU_D(X) = \sum_{X\subseteq Tj \wedge Tj\in D} TU(Tj)$.

Definition 6. Prefix Utility of an itemset X in a Transaction

$PrefixUtil_D(X,T)=\sum_{i\in PrefixSet(X,T)} u(i,T)$

Example $PrefixUtil(\{ac\},T_3)=u(a,T_3) + u(b,T_3)+ u(c,T_3) =3+12+15=30$

Definition 7. Prefix Utility of an itemset X in a Dataset D

Example PrefixUtil({ac})=PrefixUtil({ac},T_1)+PrefixUtil({ac},T_2)+ PrefixUtil({ac},T_3)=8+36+30=74

Property 1.For any itemset X in ad dataset D, the following relationship holds :

$TWU_D(X) \geq PrefixUtil_D(X) \geq u_D(X)$

Clearly, $TWU_D(X) \geq u_D(X)$.

TWU satisfies the downward closure property, thatis, for all $Y \subseteq X$, $TWU_D(Y) \geq TWU_D(X)$.Most of HUI mining methods uses the TWU values of the itemsets to prune the search space.That is, they find all the itemsets whose TWU is no less than the minimum utility threshold. Since $TWU_D(X)$ is an overestimate of $u_D(X)$, The procedure does not miss any high utiltyitemset.

Property 2. Anti-monotone Propoerty

It says that support for an itemset never exceed the support for its subsets.

For example If x& y are 2 itemsets such that $x \cap y = \emptyset$, then supp(x U y) < supp(x) and

It makes the search in the itesmet lattice easier by avoiding large number of useless cases.

Property 3. Apriori Property

Every subset of a frequent itesmet is also frequent

Table 1: Dataset

TID	ITEMSET	Tu
1	(A, 2),(B,1),(C,3),(E,2)	35
2	(A, 1),(D,1),(F,2)	8
3	(B,2),(C,1),(D,3),(E,3)	21
4	(C, 3),(F,2)	20
5	(A, 2) ,(B, 1), (D, 1) ,(E,2), (F,2)	20
6	(A, 2),(B,3),(C,2)	31

Table 2: Item with Their External Utility

Item name	A	B	C	D	E	F
External Utility	5	3	6	1	2	1

METHODOLOGY OF IFP

Tree is very useful structure to store data and process. The more data store the more memory consumes. Researcher mostly uses tree structure which is based on FP-Growth tree model. Every Research paper has slight difference in using and processing transaction and maintains tree structure.

Processing of Transaction is more important. From data stream perspective view ,data comes at very high speed, So processing must be fast and scan only ones when transaction arrive. The more task algorithm performs the more time it takes and gives more accuracy. So Algorithm has to balance between accuracy and speed.

There are many Research papers which suggest resource aware mining is very useful. As data stream contains unbounded and unlimited transaction, resource must be maintain rather than overloading and total hamper processing

which lead to faulty results.

As discuss in Introduction Chapter, There are three types of window model. Among them sliding window model is best preference if recent transaction is more important. Using Sliding window model the data are mines based on recent data, and current mining result is more important than older.

DFS and BFS are two techniques which most data mining researcher are using. Both techniques are best in certain scenario.

Here we are using two structures as shown below.

Table 3: RankItem Node Structure

RankItem Node		
1	**Name**	**Item Name**
2	UnitList[win_Size]	Store Utility of item of window size
3	TotalUnit	Total utility of window
4	TotalTrans	Total Transaction in window that contain item
5	ExternalUtility	External Utility of item
6	TotalProfit	Total Profit generated by item
7	Hot	Shows that item is recent or not

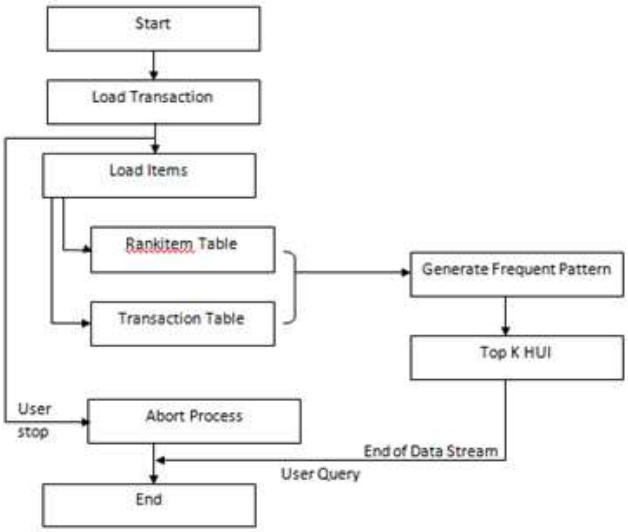
Table 4: Transaction Node Structure

Transaction Node		
1	TransNum	Transaction Number
2	ItemList	Contain Item list of transaction
3	TotalItem	Total Item transaction contain
4	TransUtility	Total Transaction Utility

As shown above Data are store in above structure. Structure is very compact but powerful in nature as it contains information of item based on that mining is performed.

In the algorithm, the user provide only two input which are ‘k’ and ‘Win_Size’. Here ‘K’ value defines top k high utility item user want. And ‘Win_Size’ define window size. In Window, Transactions are in batch wise. Each batch contains certain number of transaction.

Below is Flow chart of Algorithm.



As seen in Flowchart data comes in continuously. Here we are using sliding window model, we first load data in transaction node List. Also simultaneously add item information in rankitem node List. Whenever user requires result, the mining result shown to user and generated after each batch, it means after each batch mining is performed. Result will show to user only when user perform query or at the end of Data Stream. The reason is that producing result to screen takes more time and consumes memory and it is overload to algorithm. If user doesn't want the result right now why should display to screen.

The process is very simple yet effective. Here we are not using tree structure, because this algorithm is based on recent item and item which generate high profit is most valuable also our structure effective track item which generate higher profit in recent window. Here Pruning is simultaneously done when new transactions arrive.

Itemset which generate higher profit are sorted in Rankitem List .Based on that potential itemsets are generated.

Algorithm 1. Loading Transaction and data into Structure

Input: Win_size,K, Transaction T,Batch Bi,

Output: Top-K HUIs

```
1.  TxNum←Tnum  % Win_Size
2.  For every T ∈ Bi do
3.      Transaction[TxNum].TransNum = TxNum;
4.      For every Item  I ∈ T do
5.          Transaction[TxNum].ItemList.Add(I);
6.          Transaction[TxNum].TotalItem++;
7.          IF Item I ∈RankItem List then
8.              RankItem.UnitList [TxNum]=I.InternalUtility
9.              RankItem.ExternalUtility =I.ExternalUtility
10.             RankItem. Hot =True
11.         Else
12.             RankItem.Name=I.Name
13.             RankItem.UnitList [TxNum]=I.InternalUtility
14.             RankItem.ExternalUtility =I.ExternalUtility
15.             RankItem. Hot =True
16.         IfRankItem.totalTrans !=  Win_Size
17.             RankItem.totalTrans++;
18.     Utility= I.ExternalUtility *I.InternalUtility;
19.     Transaction[TxNum].utility +=Utility;
20.     If  it is not First Slinding window then
21.         For everyrankItem node I in List do
22.             IfrankItem node I is not hot then
23.                 IfrankItemtotalTrans != 0 then
24.                     RankItem.totalTrans--;
25.                 Else
26.                     Remove RankItem;
27.             Else
28.                 RankItem.Hot =False
29.         For everyNew Batch do
30.             Call Algorithm 2 Mining
31.             Return Top K HUI
```

In the First Algorithm, we just add data to our structure. Here RankItem list is class structure which has 7 fields and utility list field is of window size. Each item information store in rankItem class structure. So there are many item in data stream, so it create array of RankItem class.

Here TxNum is transaction number that cannot greater than window size. Each Transaction is loaded into

Transaction Class structure than added to list of transaction.

Here each item has internal and external utility. Both utility are store in RankItem class structure which is in the RankItem List

Here we store and maintain information of recent window. When Window moves or new batch arrives, RankItem node in list also update and same for transaction list. Advantages is that it remain compact in size.

In algorithm 1, for each transaction we add transaction number to Transaction node. For every item in transaction we added into transaction's itemlist field. If item found in RankItem list then we update information like its internal utility and total transaction value also hot field which says that whether it is recently arrives or not.

After First window, for every transaction we check for the item which is not hot. For that item we decrement the totalTransaction field of RankItem node .This way keep track of recent item. For each item whose value is true for hot field, it is updated to false.

Mining is performed after every batch and after first window complete.

Algorithm 2 is mining where actual work is done. In this algorithm, for every Rank item in List update Rank Item Toal Unit by summing up all the unit from Unit List field. After that Rank Item. Total Profit calculated. Here we min_supp taken as avg of all total profit by all item. Avg Total Unit is also used to filter data items. Potential is list of Rank Item nodes which generate higher profit and sorted descending order according to total profit.

After that HuiList is generated by taking each item at a time and adding second highest and possible all variation. Also side by side we calculated utility also of itemset. It generate very high Itemset and their profit. After that it check for min_Supp. If it is greater than min_Supp then added to topKHui List and after that sorting is done based on profit.

Algorithm 2. Mining

Input: Batch Bi

Output: Top-K HUIs

1. **If** it is not First Slinding window **then**
2. **For** everyNew Batch **do**
3. topKHui=∅
4. **For** every RankItem node in List **do**
5. RankItem.TotalUnit=∑ RankItem.UnitList
6. RankItem.TotalProfit=RankItem.TotalTransaction * RankItem.totalUnit * RankItem.ExternalUtility
7. Min_Supp=∑ Rankitem.ToalProfit / Total RankItem node in List
8. AvgTotalUnit=∑ Rankitem.ToalUnit / Total RankItem node in List
9. Potential=List all Rankitem node where RankItem.ToalProfit>Min_Supp and RankItem.TotalUnit>AvgTotalUnit
10. Potential = sorting all rankitem Node by decending order of Total Profit
11. HUI List = generate a set of top k HUIs by combination taking each item at ones and adding second highest profitable item and so on. And calculate utility also simultaneously
12. **For** every Itemset in Hui List **do**
13. **If** utility >Min_Supp**then**
14. Add to topKHui List
15. Sort by decending order of their profit
16. **If** K >topKHui list **then**
17. Display all Itemset with Their utility profit.

18. **Else**
19. Display top K item and prune all itemsets
20. **Return topkHUI**

If k value is higher than topKHui List then print all itemset with their utility profit otherwise display only k itemset and prune all itemset form topkItem list.

THE PERFORMANCE OF ICFP IS PRESENTED

In this chapter, the proposed method is evaluated. All the algorithm are implemented in C#. The experiments are conducted on Intel core i3 2.53 GHz computer with 3GB RAM.

For experiment, four datasets are used. They are chess, mushroom, connect4 and T10I4D100K.First three datasets were prepared by Roberto Bayardo from the UCI datasets. The Last dataset is the IBM Synthetic dataset T10I4D100K, where the numbers after T, I, and D represent the average transaction size, average size of maximal potential frequent patterns and the number of transaction, respectively. In all the dataset, External utility of item is not provided and also quantity of each item in transaction as well. Therefore External utility of item is generated using log normal distribution and quantity of item in transaction is generated randomly between 1 and 10.

Table 5: Details of Datasets

	Transaction	Items	Max object in Trans.
chess	3196	75	37
mushroom	8124	119	23
connect4	67557	129	43
T10I4D100K	100000	870	29

As the K size and winSize Varies runtime and performance also change. K value defines maximum number of top High utility pattern. For small value of k, Number of itemset store and maintain in top K HUI list are very less and therefore less memory consumption. Less memory consumption improves performance.

WinSize is also important. For high value of WinSize, High utility itemset is having more potential to generate more profit. For small value of WinSize, it gives best result based on current window size transaction only.

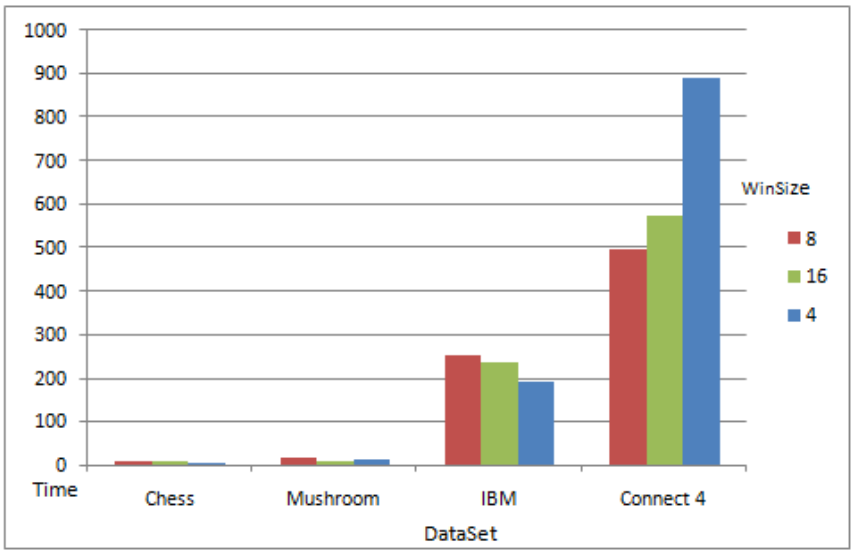


Figure 1: Window Size Effect on Runtime

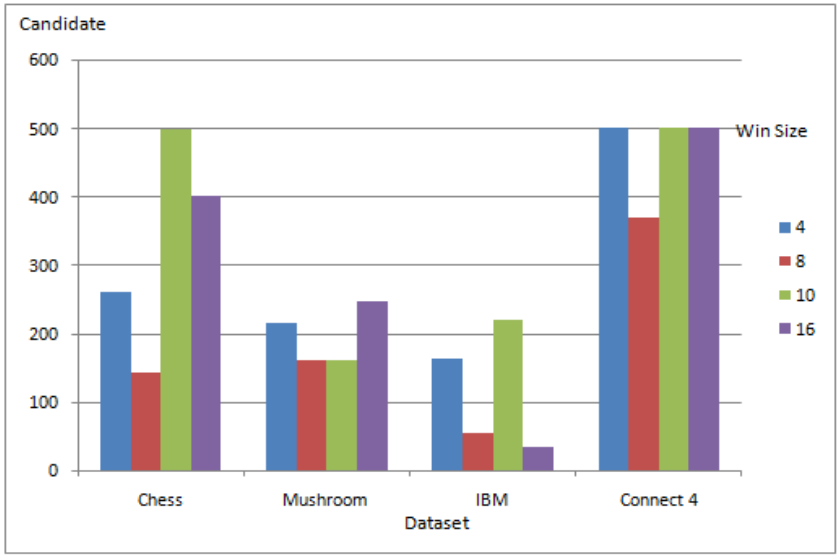


Figure 2: Window Size Effect on Candidate Itemset

As seen in above chart, Change in winSize value affects timing. Timing is also affected by number of items in RankItem list and potential Items. The more number of Potential Item the more candidate itemset generated and takes more time.

Free up RankItem node which have 0 Total transaction values reduce memory and improves performance. In this algorithm Mining is done at every batch. This is also affecting performance and timing.

CONCLUSIONS AND FUTURE WORK

Our research focuses on Designing computation and memory efficient algorithms to provide approximate results in high accuracy and confidence and developing system support help to mine useful information from data streams. Some specific research problems are identified. Result is as expected also can be improved by updating support value frequently based on window maximum and minimum value of itemset. Potential itemset generation is based on high value of frequent item. The itemset is for current window only so it can be improved by updating and maintain older itemset and prune based on time fading model. Meanwhile, the new techniques and algorithms we have developed for frequent itemset mining on streaming data are presented, and some preliminary ideas of our on-going work are discussed. Currently, we continue working on some of these problems.

REFERENCES

1. MortezaZihayat, Aijun An, Mining top-k high utility patterns over data streams, Information Sciences, Volume 285, 20 November 2014, Pages 138-161, ISSN 0020-0255
2. Doug Burdick, Manuel Calimlim and Johannes Gehrke, MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In *Proceedings of the 17th International Conference on Data Engineering*. Heidelberg, Germany, April 2001.
3. FatemehNori, MahmoodDeypir, and MohamadHadiSadreddini. 2013. A sliding window based algorithm for frequent closed itemset mining over data streams. *J. Syst. Softw.* 86, 3 (March 2013), 615-623. DOI=10.1016/j.jss.2012.10.011

4. Gangin Lee, Unil Yun, KeunHoRyu, Sliding window based weighted maximal frequent pattern mining over data streams, Expert Systems with Applications, Volume 41, Issue 2, 1 February 2014, Pages 694-708, ISSN 0957-4174

5. Bai-En Shie, Philip S. Yu, Vincent S. Tseng, Efficient algorithms for mining maximal high utility itemsets from data streams with different models, Expert Systems with Applications, Volume 39, Issue 17, 1 December 2012, Pages 12947-12960, ISSN 0957-4174,

6. Unil Yun, Gangin Lee, and KeunHoRyu. 2014. Mining maximal frequent patterns by considering weight conditions over data streams. *Know.-Based Syst.* 55 (January 2014), 49-65. DOI=10.1016/j.knosys.2013.10.011

7. MahmoodDeypir, Mohammad HadiSadreddini, and SattarHashemi. 2012. Towards a variable size sliding window model for frequent itemset mining over data streams. *Comput. Ind. Eng.* 63, 1 (August 2012), 161-172. DOI=10.1016/j.cie.2012.02.008

8. Syed KhairuzzamanTanbeer, ChowdhuryFarhan Ahmed, Byeong-SooJeong, and Young-Koo Lee. 2008. Efficient frequent pattern mining over data streams. In Proceedings of the 17th ACM conference on Information and knowledge management (CIKM '08). ACM, New York, NY, USA, 1447-1448.

9. Pauray S. M. Tsai. 2009. Mining frequent itemsets in data streams using the weighted sliding window model. Expert Syst. Appl. 36, 9 (November 2009), 11617-11625.

10. Hua-Fu Li, Suh-Yin Lee, Mining frequent itemsets over data streams using efficient window sliding techniques, Expert Systems with Applications, Volume 36, Issue 2, Part 1, March 2009, Pages 1466-1477, ISSN 0957-4174

11. Younghee Kim, Wonyoung Kim and Ungmo Kim, "Mining Frequent Itemsets with Normalized Weight in Continuous Data Streams," Journal of Information Processing Systems, vol. 6, no. 1, pp. 79~90, 2010

12. Jing Guo, Peng Zhang, Jianlong Tan, and Li Guo. 2011. Mining frequent patterns across multiple data streams. In Proceedings of the 20th ACM international conference on Information and knowledge management (CIKM '11), Bettina Berendt, Arjen de Vries, Wenfei Fan, Craig Macdonald, IadhOunis, and Ian Ruthven (Eds.).

13. AnushreeGoutamRingne, DeekshaSood, and DurgaToshniwal, "Compression and Privacy Preservation of Data Streams using Moments," International Journal of Machine Learning and Computing vol. 1, no. 5, pp.473-478, 2011.

14. Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03).

15. Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. 2003. A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28, 1 (March 2003), 51-55.

16. Motwani, R; Manku, G.S (2002). "Approximate frequency counts over data streams". VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases: 346–357.

17. Boutsis, I.; Kalogeraki, V., "Resource management using pattern-based prediction to address bursty data streams," Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2013 IEEE 16th International Symposium on , vol., no., pp.1,8, 19-21 June 2013
18. HebaTallah Mohamed Nabil, Ahmed SharafEldin and Mohamed Abd El-Fattah Belal, “Mining Frequent Itemsets from Online Data Streams: Comparative Study” International Journal of Advanced Computer Science and Applications(IJACSA), 4(7), 2013
19. Dr. S. Vijayarani, Ms. P. Sathya “An Efficient Algorithm for Mining Frequent Items in Data Streams”,in International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 3, May 2013