ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 5, July 2012

ESW-FI: An Improved Analysis of Frequent Itemsets Mining

K Jothimani, Dr Antony SelvadossThanamani

***Abstract*—Frequent itemsets mining play an essential role in many datamining tasks. The frequent itemset mining over data streams is to find an approximate set of frequent itemsets in transaction with respect to a given support and threshold. It should support the flexible trade-off between processing time and mining accuracy. It should be time efficient even when the user-specified minimum support threshold is small. The objective was to propose an effective algorithm which generates frequent patterns in a very less time. Our approach has been developed based on improvement and analysis of MFI algorithm. In this paper, we introduce a new algorithm ESW-FI, to maintain a dynamically selected set of item sets over a sliding window. We keep some advantages of the previous approach and resolve the drawbacks, and produce the improved runtime and memory consumption. The proposed algorithm gave a guarantee of the output quality and also a bound on the memory usage.**

.

***Index Terms*—Data stream, data-stream mining, Efficient Window, frequent itemset and sliding window.**

.

I. Introduction

Data stream is an ordered sequence of elements that arrives in timely order. In many application domains, data is presented in the form of data streams which originate at some endpoint and are transmitted through the communication channel to the central server. Different from data in traditional static datasets, data streams are continuous, unbounded, usually come with high speed and have a data distribution that often changes with time [1][2][3]. It is often refer to as streaming data..

Some well-known examples include market basket, traffic signals, web-click packets, ATM transactions, and sensor networks. In these applications, it is desirable that we obtain some useful information, like patterns occurred frequently, from the streaming data, to help us make some advanced decision[4]. Data-stream mining is such a technique that can find valuable information or knowledge from a great deal of primitive data.

Recently, the data stream, which is an unbounded sequence of data elements generated at a rapid rate, provides a dynamic environment for collecting data sources. It is likely that the embedded knowledge in a data stream will change quickly as time goes by[5]. Therefore, catching the recent trend of data is an important issue when mining frequent itemsets from data streams. Although the sliding window model proposed a good solution for this problem, the appearing information of the patterns within the sliding window has to be maintained completely in the traditional approach[11].

We classify the stream-mining techniques into two categories based on the window model that they adopt in order to provide insights into how and why the techniques are useful. First, each element in the datastream can be examined only once or twice, making traditional multiple-scan approaches infeasible. Second, the consumption of memory space should be confined in a range, despite that data elements are continuously streaming into the local site. Third, notwithstanding the data characteristics of incoming stream may be unpredictable; the mining task should proceed normally and offer acceptable quality of results. Fourth, the latest analysis result of the data stream should be available as soon as possible when the user invokes a query.

In this paper we consider mining recent frequent itemsets in sliding windowsover data streams and estimate their true frequencies, while making only one passover the data. In our design, we actively maintain potentially frequent itemsets ina compact data structure [8][9]. Compared with existing algorithms, our algorithm hastwo contributions as follows:

1. It is a real one-pass algorithm. The obsolete transactions are not required whenthey are removed from the sliding window.
2. Flexible queries based on continuous transactions in the sliding window can be answered with an error bound guarantee.

In this paper, we propose a remarkable approximating method for discovering frequent itemsets in a transactional data stream under the sliding window model. In our design, we actively maintain potentially frequent itemsets ina compact data structure. It is a real one-pass algorithm. The obsolete transactions are not required whenthey are removed from the sliding window. Flexible queries based on continuous transactions in the sliding window can beanswered with an error bound guarantee.

II. Preliminaries

A. Related Work

Frequent-pattern mining has been studied extensively indata mining, with many algorithms Proposed and implemented. Frequent pattern mining and its associatedmethods have been popularly used in association

K. Jothimani is with the Department of Computer Science, NGM College, Pollachi,Tamilnadu,India. Email: jothi1083@yahoo.co.in

Dr. Antony SelvadossThanamani is with the Department of Computer Science, NGM College, Pollachi. Tamilnadu, India.

Email: selvdoss@yahoo.com

.

rule mining, sequential pattern mining, structured pattern mining, iceberg cube computation, cube gradient analysis, associative classification, frequent pattern-basedclustering [26], and so on. There are a number of research works which study the problem of data-stream mining in the first decade of 21st century. Among these studies, *Lossy Counting* [3] is the most famous method of mining *frequent itemsets* (FIs) through data streams under the landmark window model. Besides the user-specified *minimum support threshold* (*ms*), Lossy Counting also utilizes an *error parameter*, ε , to maintain those infrequent itemsets having the potential to become frequent in the future. With the use of ε , when an itemset is newly found, Lossy Counting knows the upper bound of counts that itemsets may have (in the previous stream data) before it has been monitored by the algorithm.

B. Previous Algorithms

Based on the ε mechanism of Lossy Counting, [4] proposed the *sliding window* method, which can find out frequent itemsets in a data stream under the sliding window model with high accuracy. The sliding window method processes the incoming stream data transaction by transaction.[11][17] Each time when a new transaction is inserted into the window, the itemsets contained in that transaction are updated into the data structure incrementally. Next, the oldest transaction in the original window is dropped out, and the effect of those itemsets contained in it is also deleted. The sliding window method also has a periodical operation to prune away unpromising itemsets from its data structure, and the frequent itemsets are output as mining result whenever a user requests.

In the sliding window model, there are two typical mining methods: *Moment* [5] and *CFI-Stream* [6]. Both the two methods aimed at mining *Closed Frequent Itemsets* (CFIs), a complete and non-redundant representation of the set of FIs. Moment uses a data structure called *CET* to maintain a dynamically selected set of itemsets, which includes CFIs and itemsets that form a boundary between CFIs and the rest of itemsets. The CFI-Stream algorithm, on the other hand, uses a data structure called *DIU tree* to maintain nothing other than all *Closed Itemsets* over the sliding window. The current CFIs can be output anytime based on any *ms*specified by the user.

Besides, there are still some interesting research works [9] [10] [11] on the sliding window model. In [9] a false-negative approach named *ESWCA* was proposed. By employing a progressively increasing function of ms, ESWCA greatly reduces the number of potential itemsets and would approximate the set of FIs over a sliding window. In [10] a data structure called *DSTree*was proposed to capture information from the streams. This tree captures the contents of transactions in a window, and arranges tree nodes according to some canonical order.According to the overview, one crucial problem is to efficiently compute the inclusion between two itemsets. This costly operation could easily be performed when considering a new representation for items in transactions. From now, each item is represented by a unique prime number.

A sliding window over a data stream is a bag of last N elements ofthe stream. There are two variants of sliding windows based on whether N is fixed(fixed-sized sliding windows) or variable (variable-sized sliding windows). Fixed-sized windows are constrained to perform the insertions and deletions in pairs, exceptin the beginning when exactly N elements are inserted without a deletion. Variable-sized windows have no constraint[18].

Fixed-sized and variable-sized windows model several variants of sliding windows. For example, tuple-based windows correspond to fixed-sized windows, andtime-based windows correspond to variable-sized windows.

In recent two years, a new kind of data-stream mining method named *DSCA* has been proposed [12]. DSCA is an approximate approach based on the application of the *Principle of Inclusion and Exclusion* in *Combinatorial Mathematics* [7]. One of the most notable features of DSCA is that it would approximate the count of an arbitrary itemset, through an equation (i.e., Equation (4) in [12]), by only the sum of counts of the first few orders of its subsets over the data stream. There are also two techniques named *counts bounding* and *correction*, respectively, integrated within DSCA. The concept of *Inclusion and Exclusion Principle* [7] is valuable that it may also be applied in mining FIs under different window models other than the landmark window. Based on the theory of *Approximate Inclusion–Exclusion* [8], we devise and propose a new algorithm, called *ESW-FI*, to discover FIs over the sliding window in a transactional data stream.

III. PROBLEM DESCRIPTION

Let $I = \{x_1, x_2, \ldots, x_z\}$ be a set of items (or attributes). An itemset (or a pattern) X is a subset of I and written as $X = x_i x_j \ldots x_m$. The length (i.e., number of items) of an itemset X is denoted by $|X|$. A transaction, T, is an itemset and T supports an itemset, X, if $X \subseteq T$. A transactional data stream is a sequence of continuously incoming transactions. A segment, S, is a sequence of fixed number of transactions, and the size of S is indicated by s. A window, W, in the stream is a set of successive w transactions, where $w \geq s$. A sliding window in the stream is a window of a fixed number of most recent w transactions which slides forward for every transaction or every segment of transactions. We adopt the notation I_l to denote all the itemsets of length l together with their respective counts in a set of transactions (e.g., over W or S). In addition, we use T_n and S_n to denote the latest transaction and segment in the current window, respectively. Thus, the *current window* is either $W = \langle T_{n-w+1}, \ldots, T_n \rangle$ or $W = \langle S_{n-m+1}, \ldots, S_n \rangle$, where w and m denote the size of W and the number of segments in W, respectively.

In this research, we employ a *prefix tree* which is organized under the *lexicographic order* as our data structure, and also processes the growth of itemsets in a lexicographic-ordered away. As a result, an itemset is treated a little bit like a *sequence* (while it is indeed an itemset). A superset of an itemset X is the one whose length is above $|X|$ and has X as its *prefix*. We define $\text{Growth}_l(X)$ as the set of supersets of an itemset X whose length are l more than that of X, where $l \geq 0$. The number of itemsets in $\text{Growth}_l(X)$ is denoted by $|\text{Growth}_l(X)|$.

We adopt the symbol cnt(X) to represent the count-value (or just count) of an itemset X. The count of X over W, denoted as $\text{cnt}_W(X)$, is the number of transactions in W that support X. So $\text{cnt}_S(X)$ represents the count of X over a segment S. Given a user-specified *minimum support threshold* (*ms*), where $0 < ms \leq 1$, we say that X is a *frequent itemset* (FI) over W if $\text{cnt}_W(X)$ $ms \times w$, otherwise X is an *infrequent itemset* (IFI). The FI and IFI over S are defined similarly to those for W.

Given a data stream in which every incoming transaction has its items arranged in order, and a *changeable value* of

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 5, July 2012

*ms*specified by the user, the problem of mining FIs over a sliding window in the stream is to find out the set of frequent itemsets over the window at different slides.

We remark that most of the existing stream mining methods[13] [14] [15] [19] work with a basic hypothesis that they know the user-specified *ms*in advance, and this parameter will remain unchanged all the time before the stream terminates. This hypothesis may be unreasonable, since in general, a user may wish to tune the value of *ms*each time he/she makes a query for the purpose of obtaining a more preferable mining result. An unchangeable *ms*leads to a serious limitation and may be impractical in most real-life applications. As a result, we relax this constraint in our problem that the user is allowed to change the value of *ms*at different times, while our method must still work normally.

Using the original count-values of subsets to approximate the count of X may sometimes bring about considerable error. For a better approximation, there is a possible way, which is to *bound*the range of counts of subsets for the itemset to be approximated.

Let Y be a 3-itemset (to be approximated) and y be a 1-subset of Y. To obtain a better approximation of Y, the count-value to each subset y with respect to Y has a particular range, which can be determined by Y's 2-subsets that have y as their common subset, respectively. This range of y's count is bounded by an *upper bound* (i.e., the maximum) and a *lower bound* (i.e., the minimum), and count-values within this range are the set of portions of y's original count which is more relevant for y with respect to Y[21]. We define $\mathrm{Scb}_y(Y)$ as the set of count-values of y with respect to Y within the range obtained through the aforesaid manner. Besides, the upper bound and the lower bounds of count-values of y are denoted by $\mathrm{ub}_y(Y)$ and $\mathrm{lb}_y(Y)$, respectively.

Lemma 1 *Let* $\mathrm{cnt}_{ub}(Y)$ *and* $\mathrm{cnt}_{lb}(Y)$ *respectively be the approximate counts of Y obtained by using* $\mathrm{ub}_y(Y)$ *and* $\mathrm{lb}_y(Y)$*of count-values to every 1-subset* $y \in Y$ *during the approximating process. Then* $\mathrm{cnt}_{ub}(Y) \leq \mathrm{cnt}_{lb}(Y)$.

Proof: Let T_{ub} and T_{lb}be the sums of counts of 1-subsets of Y obtained by choosing $\mathrm{ub}_y(Y)$ and $\mathrm{lb}_y(Y)$ for each $y \in Y$, respectively. Then $T_{ub} \geq T_{lb}$since $\mathrm{ub}_y(Y) \geq \mathrm{lb}_y(Y)$ for each y. According to (1) with the parameters setting m=3 and k=2, we can eventually obtain the following simplified equation: $\mathrm{cnt}(Y) \doteq (1 - \alpha_2^{23})\, c + (\alpha_1^{23} - 1)d$, where the symbol $\alpha_j^{k\ md}$ denotes the coefficient of linearly transformed *Chebyshev polynomial*, and c and d represent the sum of counts of Y's 2-subsets and that of counts of Y's 1-subsets, respectively. Since the value of α_2^{23} is less than 1, the coefficient

($\alpha_1^{23} - 1$) for 1-subset term is then negative, which means that the value of 1-subset term will be subtracted from the other term. Thus, by substituting $T_{ub}$and $T_{lb}$for *d*respectively in the approximate equation and knowing that $T_{ub} \geq T_{lb}$, we have cntub(Y) ≤cntlb(Y).

IV. EFFICIENT SLIDING-WINDOW PROCESSING

In research works under the sliding window model [4] [5] [6], the sliding of window is handled transaction by transaction; however, we have a different opinion. Unlike the landmark window model, transactions in the sliding window model will be both inserted into and dropped out from the window. The transaction-by-transaction sliding of a window leads to excessively high frequency of processing. In addition, since the transit of a data stream is usually at a high speed, and the impact of one single transaction to the entire set of transactions (in the current window) is very negligible, making it reasonable to handle the window sliding in a wider magnitude. Therefore, for an incoming transactional data stream to be mined, we propose to process on a *Segment-oriented window sliding*.

We conceptually divide the sliding window further into several, say, m, segments, where the term *segment* is the one we have defined earlier in Section 3. Each of the m segments contains a set of successive transactions and is of the same size s (i.e., contains the equal number of s transactions). Besides, in each segment, the summary (which contains I_1, I_2, and the *fair-cutters* which we will introduced later) of transactions belonging to that segment is stored in the data structure we use. We call the sliding of window "segment in-out," which is defined as follows.

Definition 1 (**Segment in-out**) Let$S_c$denote the current segment which is going to be inserted into the window next (after it is full of s transactions). A *segment in-out* operation (of the window) is that we first insert $S_c$into and then extract S_{n-m+1} from the original window, where n denotes the id of latest segment in the original window. Therefore, the windows before and after a sliding are $W = <S_{n-m+1}, \ldots, S_n>$ and $W = <S_{n-m+2}, \ldots, S_n, S_c>$, respectively.

By taking this segment-based manner of sliding, each time when a segment in-out operation occurs, we delete (or drop out) the earliest segment, which contains the summary of transactions of that segment, from the current window at each sliding. As a result, we need not to maintain the whole transactions within the current window in memory all along to support window sliding[24][25]. In addition, we remark that the parameter m directly affects the consumption of memory. A larger value of m means the window will slide (update) more frequently, while the increasing overhead of memory space is also considerable. In our opinion, an adequate size of m that falls in the range between 5 and 20 may be suitable for general data streams.[20]

Theorem 1 *For a 2-itemset X and a threshold ms, let* $\mathrm{TP}_{ub}(X)$ *and* $\mathrm{TP}_{lb}(X)$ *be the true-positive rates of X's 3-supersets in the mining result resulting from adopting Ubc and Lbc to all the 1-subsets of each superset, respectively. Then we have* $\mathrm{TP}_{ub}(X) \leq \mathrm{TP}_{lb}(X)$.

Proof Let P be the number of FIs of X's 3-supersets with respect to *ms*. Also, let P_{ub} and P_{lb}respectively be the numbers of true FIs of X's 3-supersets found by choosing Ubc and Lbc to 1-subsets. According to Lemma 1, we have $\mathrm{cnt}_{ub}(Y) \leq \mathrm{cnt}_{lb}(Y)$ for each 3-superset Y of X, which means that the number of true-positive itemsets (i.e., FIs) obtained by choosing Lbc is at least equal to that of choosing Ubc, i.e., $P_{ub} \leq P_{lb}$. Since $\mathrm{TP}_{ub}(X) = P_{ub}/P$ and $\mathrm{TP}_{lb}(X) = P_{lb}/P$, we then have $\mathrm{TP}_{ub}(X) \leq \mathrm{TP}_{lb}(X)$.

Theorem 2 *For a 2-itemset X and a threshold ms, let* $\mathrm{TN}_{ub}(X)$ *and* $\mathrm{TN}_{lb}(X)$ *be the true-negative rates of X's 3-supersets in the mining result resulting from adopting Ubc and Lbc to all the 1-subsets of each superset, respectively. Then we have* $\mathrm{TN}_{ub}(X) \geq \mathrm{TN}_{lb}(X)$.

Proof Let N be the number of IFIs of X's 3-supersets with respect to *ms*. Also, let N_{ub} and N_{lb}respectively be the numbers of true IFIs of X's 3-supersets determined by choosing Ubc and Lbc to 1-subsets. According to Lemma 1, we have $cnt_{ub}(Y) \leq cnt_{lb}(Y)$ for each 3-superset Y of X, which means that the number of true-negative itemsets (i.e., IFIs) determined by choosing Ubc is at least equal to that of choosing Lbc, i.e., $N_{ub} \geq N_{lb}$. Since $TP_{ub}(X) = N_{ub}/N$ and $TP_{lb}(X) = N_{lb}/N$, we then have $TP_{ub}(X) \geq TP_{lb}(X)$.

Now we discuss the issue of selecting suitable count-values in the bounded range of subsets for approximating an itemset. According to Lemma 1 in Section 3, using the *lower bound of counts* (Lbc) for 1-subsets always results in a higher approximate count for an itemset than that of using the *upper bound of counts* (Ubc). It follows from Theorem 1 and Theorem 2 that, adopting Lbc to the 1-subsets to approximate the 3-supersets of an 2-itemset X will reach a higher *true-positive rate* (i.e., *recall ratio*) in the result than that of using Ubc, while choosing Ubc to the 1-subsets will obtain a higher *true-negative rate* (which usually concerns a higher *precision ratio*) in the result than that of using Lbc. However, it is actually unknown whether to adopt Ubc or Lbc to1-subsets for approximating an itemset Y.

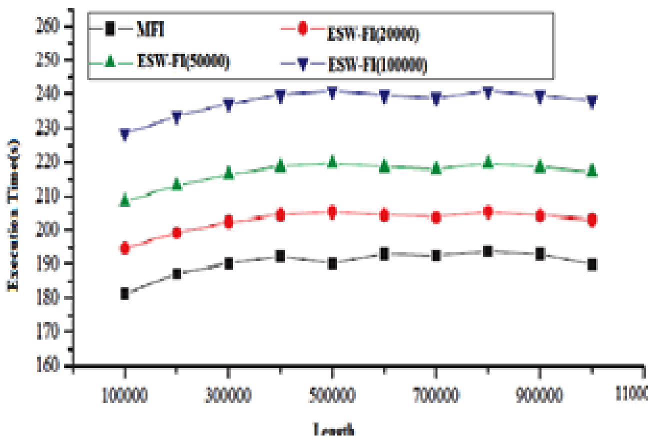
V. Experimental Result

We performed extensive experiments to evaluate the performance of our algorithmand we present results in this section. We compared our algorithm (ESW-FI)with the MFI algorithm [7].Then we tested the adaptability of ESW-FI by changing the data distribution ofthe dataset.

The experiments were performed on a Pentium processor with 4G memory,running Windows 2007 (SP4). Our algorithm is implemented in C# and compiledby using Microsoft Visual Studio 2008. In the implementation of three algorithms, weused the same data structures and subroutines in order to minimize the performancedifferences caused by minor differences.

We have usedT20 data set in the experiments. The T20 dataset is generated by the IBM data generator [1]. For T20, the average size of transactions, the average size of the maximal potential frequent itemsets and the numberof items are 20, 4, and 1 000, respectively.

Fig 5.1: Execution time of MFI Vs ESW-FI for T20



The data sets were broken into batches of10K size transactions and provided to our program through standard input.For ESW-FI, the whole procedure can be divided into two different phases namely, window initialization phase, which is activated when the number of transactions generated so far is not more than the size of the sliding window and windowsliding phase, which is activated when the window is full of generated transactions.

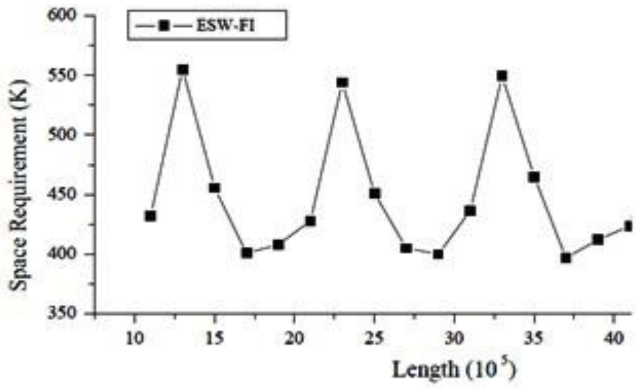


Fig 5.2: Space Requirement of ESW-FI with T20

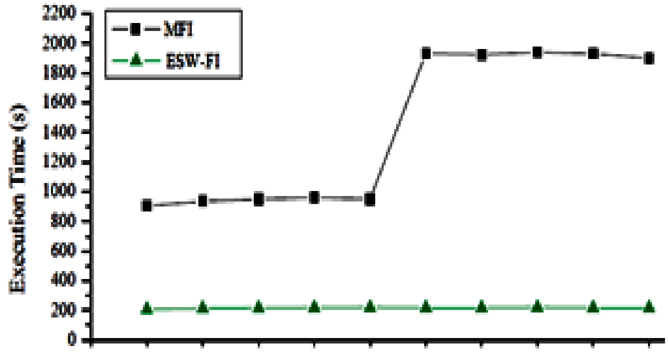


Fig 5.3: Execution time of MFI Vs ESW-FI for Updated Transactions

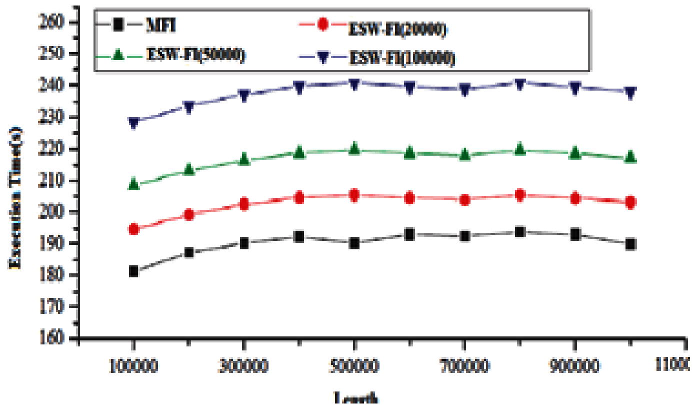


Fig 5.4: Space Requirement between MIF Vs ESW-FI with more number of transactions

Comparison between MFI and ESW-FI basedon different block size to observe the influence of different block sizes on algorithm, we take 2×104, 5×104and 10 × 104 as an example. Meanwhile, the size of the window is fixed 5 × 105.In this case, this window contains 25, 10 and 5 blocks, respectively.

The experimental results of the execution time and the space requirement are plotted in Figure 5.1 to 5.4, respectively. We collected the total number of seconds and the size of storage space required in KB per 10 × 5 transactions. These results basically keep stable as the window moves forward. The above experimental results provide evidence that two algorithm can handle long data streams both. As ESW-FI may keep multiple triples for one significant itemset, it needs more memory and time than MFI does.

VI. CONCLUSION

We mainly discuss how to discover recent frequent itemsets in sliding windows over data streams. An efficient algorithm is presented in detail. Compared with previous algorithms, this algorithm doesn't keep the data in the window, which considerably increase the scalability of the algorithm. Moreover, it can figure out answers with an error bound guarantee for continuous transactions in the window. The extensive experiment results demonstrate the effectiveness and efficiency of our approach. We can implement this algorithm in traffic networks especially for mining the itemsets in high speed streams.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of the 1993 International Conference on Management of Data, pp. 207-216, 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487-499, 1994

[3] J.H. Chang & W.S. Lee, *"A sliding window method for finding recently frequent itemsets over online data streams'*, Journal of Information Science and Engineering, 20(4), 2004, pp. 753–762

[4] Y. Zhu & D. Shasha, *"Stat Stream: statistical monitoring of thousands of data streams in real time*", Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 358–369.

[5] G.S. Manku& R. Motwani, *"Approximate frequency counts over data streams"*, Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 346–357.

[6] J.H. Chang & W.S. Lee, *"A sliding window method for finding recently frequent itemsets over online data streams"*, Journal of Information science and Engineering, 20(4), 2004, pp. 753–762.

[7] M.N. Garofalakis, J. Gehrke, & R. Rastogi, Querying and mining data streams: you only get one look (A Tutorial), Proc. 2002 ACM SIGMOD Conf. on Management of Data, Madison, Wisconsin, 2002, p. 635.

[8] Y. Zhu & D. Shasha, *"Stat Stream: statistical monitoring of thousands of data streams in real time*", Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 358–369.

[9] G.S. Manku& R. Motwani, *"Approximate frequency counts over data streams"*, Proc. 28th Conf. on Very Large Data Bases, Hong Kong, China, 2002, pp. 346–357.

[10] J.H. Chang & W.S. Lee, *"A sliding window method for finding recently frequent itemsets over online data streams"*, Journal of Information science and Engineering, 20(4), 2004, pp. 753–762.

[11] J. Cheng, Y. Ke, & W. Ng*," Maintaining frequent itemsets over high-speed data streams"*, Proc. 10th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Singapore, 2006, pp.462–467.

[12] C.K.-S. Leung & Q.I. Khan, *"DSTree: a tree structure for the mining of frequent sets from data streams,"* Proc. 6th IEEE Conf. on Data Mining, Hong Kong, China, 2006, pp. 928–932.

[13] B. Mozafari, H. Thakkar, & C. Zaniolo, "*Verifying and mining frequent patterns from large windows over data streams",* Proc. 24th Conf. on Data Engineering, Mexico, 2008, pp. 179–188.

[14] K.-F. Jea& C.-W. Li, "Discovering frequent itemsets over transactional data streams through an efficient and stable approximate approach, Expert Systems with Applications", 36(10), 2009, pp. 12323–12331.

[15] F. Bodon, *"A fast APRIORI implementation"*, Proc. ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03), 2003.

[16] N. Jiang and L. Gruenwald, *"Research Issues in Data Stream Association Rule Mining*". In SIGMOD Record, Vol. 35, No. 1, Mar. 2006.

[17] Frequent Itemset Mining Implementations Repository (FIMI).Available: http://fimi.cs.helsinki.fi/

[18] Y. Chi, H. Wang, P.S. Yu, & R.R. Muntz, *"Moment: maintaining closed frequent itemsets over a stream sliding window"*, Proc. 4th IEEE Conf. on Data Mining, Brighton, UK, 2004, pp. 59–66.

[19] N. Jiang & L. Gruenwald, *"CFI-Stream: mining closed frequent Itemsets in data streams"*, Proc. 12th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, Philadelphia, PA,USA, 2006, pp. 592–597.

[20] Quest Data Mining Synthetic Data Generation Code. Available: http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html

[21] P. Indyk, D. Woodruff, *"Optimal approximations of the frequency moments of data streams"*, Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp.202–208, 2005.

[22] H.F Li, S.Y. Lee, M.K. Shan, *"An Efficient Algorithm for Mining Frequent Itemsets over the Entire History of Data Streams*", In Proceedings of First International Workshop on Knowledge Discovery in Data Streams 9IWKDDS, 2004.

[23] H.F Li, S.Y. Lee, M.K. Shan, *"Online Mining (Recently) Maximal Frequent Itemsets over Data Streams*", In Proceedings of the 15th IEEE International Workshop on Research Issues on Data Engineering (RIDE), 2005.

[24] Lin C.-H., Chiu D.-Y., Wu Y.-H. and Chen A.L.P.: Mining Frequent Itemsets from Data Streams with a Time-Sensitive Sliding Window. In Proc SIAM International Conference on Data Mining. (2005).

[25] Ho, C. C., Li, H. F., Kuo, F. F., & Lee, S. Y. (2006). Incremental mining of sequential patterns over a stream sliding window. In Proceedings of IEEE international workshop on mining evolving and streaming data.

[26] K Jothimani, Dr Antony SelvadossThanmani, "MS: Multiple Segments with Combinatorial Approach for Mining Frequent Itemsets Over Data Streams", IJCES International Journal of Computer Engineering Science, Volume 2 Issue 2 ISSN : 2250:3439

[27] K Jothimani, Dr Antony SelvadossThanmani, "An Algorithm for Mining Frequent Itemsets," "International Journal for Computer Science Engineering and Technology IJCSET, March 2012,Vol 2, Issue 3,1012-1015.

K Jothimanireceived her Bachelor degree in Computer Science from Bharathidasan University, Trichy in 2003. She received her Master degree in Computer Applications in 2008. She pursued her Master of Philosophy in Computer Science in the year 2009 from Vinayaka Missions University, Salem. Currently she is a research scholar of the Department of Computer Science, NGM College under Bharathiyar University, Coimbatore. She had five years of experience in the computer field in technical as well as non-technical. She is a life member of Indian Society for Technical Education from the year 2009. Also she is active member of Computer Society of India (CSI). She has published more than ten papers in international and national conferences including international journals. Her area of interests includes Data mining, Knowledge Engineering and Image Processing.

Email: jothi1083@yahoo.co.in

.

Dr Antony SelvadossThanamaniis presently working as professor and Head, Dept of Computer Science, NGM College, Coimbatore, India(affiliated to Bharathiar University, Coimbatore). He has published many papers in international/national journals and written many books. His areas of interest include E-Learning, Software Engineering, Data Mining, Networking, Parallel and Distributed Computing. He has to his credit 25 years of teaching and research experience. His current research interests include Grid Computing, Cloud Computing, Semantic Web. He is a life member of Computer Society of India, Life member of Indian Society for Technical Education, Life member of Indian Science Congress, Life member of Computer Science, Teachers Associates, Newyork.
Email:-selvdoss@yahoo.com