International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:8, 2011

Content Based Sampling over Transactional Data Streams

Mansour Tarafdar, Mohammad Saniee Abade

***Abstract*—**This paper investigates the problem of sampling from transactional data streams. We introduce CFISDS as a content based sampling algorithm that works on a landmark window model of data streams and preserve more informed sample in sample space. This algorithm that work based on closed frequent itemset mining tasks, first initiate a concept lattice using initial data, then update lattice structure using an incremental mechanism.Incremental mechanism insert, update and delete nodes in/from concept lattice in batch manner. Presented algorithm extracts the final samples on demand of user. Experimental results show the accuracy of CFISDS on synthetic and real datasets, despite on CFISDS algorithm is not faster than exist sampling algorithms such as Z and DSS.

***Keywords*—**Sampling, Data Streams, Closed Frequent Itemset Mining

I. Introduction

INCREMENTAL mining on data streams is one of the most interesting research issues of data mining in recent years. Data streams have the specific features such as: rapid incoming rate, unbounded amount of input data, limitation in main memory usage, one step scanning process, uncontrolled order of arrival data. Due to these reasons, it's necessary to use a reduction process to reduce the size of data streams. Sampling is one of the effective methods to reduce the cost of stream mining process. But, Simple random sampling or its similar reservoir sampling in streaming data has a few weaknesses. An SRS sample may not sufficiently represent the content of dataset due to random fluctuation in the sampling process. This difficulty is particularly apparent at small sample ratios which are the case for very large databases with limited memory.Several sampling algorithms have been presented for sampling over data streams. The first algorithm was introduced by Vitter [1] in 1985. The algorithm presented in this paper requires one data scanning and the samples are kept in fixed space. High speed was one of the advantages of this algorithm and information loss sampling process was one of its disadvantages.Counting and concise [2], Sample-and-hold [3], Sticky sampling [4] and Probabilistic-Inplace [5] that presented for sampling over stream of data, keep information of data statistically and algorithmically to sample based on content of data.

M. Tarafdar is MS student in Islamic Azad University, Qazvin branch, Qazvin, Iran (e-mail: tarafdar.mansour@gmail.com).

M. S. Abadeh is Assistant Professor of Faculty of Electrical and Computer Engineering ,Tarbiat Modares University, Jalal Ale Ahmad Highway, Tehran, Iran (P.O.Box: 14115-143 ,Phone: +98 21 8288 4349, Fax: +98-21-82884325, e-mail: saniee@modares.ac.ir)

In 2006, DSS Algorithm [6] was introduced which took samples based on contents of transactional data streams. In this algorithm efforts were made to approach the sampling distance to data stream space. Calculations have shown that this algorithm is more efficient in discovering of frequent items in normal and noisy data [7] than Z [1] and LCA [4] Algorithms but It suffers from low speed execution. The author shows the proposed sampling method is accurate than approximate counting for frequent itemset mining task. Frequent itemset mining (FIM) over streaming data is popular particularly among researchers of data mining. However, mining the complete set of frequent itemsets in data streams is practically impracticable in some cases. To efficiently solve this problem, closed frequent itemsets was focused on as condensed illustration. An itemset is closed if none of its super itemsets has the equal support value with it, and a CFI is both closed and frequent. Many researches on CFI mining over static dataset have been proposed such as Closet[8], Closet+[9] , Charm[10], DCI-Closed [11] , FP-Close [12], LCM [13] and Recently, some CFI mining approaches over stream's sliding window were presented include Stream-Close[14] , Moment+[15] and Moment[16]. Although these algorithms taking advantages from efficient data structure and mechanism, the obtained CFI is not acceptable set of entire of data stream because of using sliding window. On the other hand, the mining process over whole dataset is extra time consuming for each run and minimum support value.In this paper, due to massive size of data streams and the complexity of closed frequent itemset mining tasks over data streams, we propose CFISDS to speed-up mining process using various minimum support values and improve the efficiency of closed frequent itemset mining algorithm results over transactional data streams. CFISDS is the first closed concept based sampling algorithm that initiate a concept lattice as synopsis data structure and update the concept lattice in batch manner. The algorithm is based on sound mathematical foundation of Formal Concept Analysis and stores the closed itemsets in a lattice based synopsis. CFISDS utilize landmark window model over data streams and extract samples on demand of user. Experimental results show the accuracy of CFISDS on synthetic and real datasets.The rest of this paper is organized as follows. Definition of the problem describe in section 2. In section 3, we discuss the proposed method and we introduce a new approach to evaluate our sampling algorithm in Section 4. We report our experimental results in Section 5 and conclude our work in Section 6.

II. Problem Definition

Suppose $I = \{i_1, i_2, \dots, i_m\}$ is a set of items. One transaction in the form of $T = \{Tid, x_1, x_2, \dots, x_n\}$ in which $x_i \in I$, $1 \leq i \leq n$ is defined. The number n shows the transaction length

and Tid is unique identifier of transaction. An itemset that contain k items called k-itemset that includes (k-1)-itemsets, (k-2)-itemsets… 2-itemsets and 1-itemsets. In transactional data stream TDS $= \{T_1, T_2, \dots, T_N\}$, TDS is a continuous stream and T_N is the last incoming transaction. According to the definition for an itemset, frequent itemset is defined as follows. Itemset X is a frequent itemset if can obtain support value more than user defined minimum support. The support value of itemset X is equal to the number of transactions that contain itemset X as a sub itemset. In this case we define absolute minimum support value for support checking process.

Itemset X is called a closed frequent itemset if (a) it is frequent and (b) there's no proper superset like Y for X such that support value x and y are equal. Information lossless result and non-redundant Itemset X is called a closed frequent itemset if (a) it is frequent and (b) there's no proper superset like Y for X such that support value X and Y are equal.

A formal concept, which is show by triple(T, O, I), contain two sets T (itemsets) and O (properties) and the relation I between two sets T and O. For a set $X \subseteq T$ of itemsets, common properties sets to the objects in X is defined as $X' = \{o \in O | tIo\ for\ all\ t \in X\}$, and for set $Y \in O$ of properties, the set of itemsets common to the properties in Y is defined as $Y' = \{t \in T | tIo\ for\ all\ o \in Y\}$. A formal concept of the context (T, O, I) will be showed by pair(X, Y) in which $X \subseteq T,\ Y \in O\ ,\ X' = Y$. In concept (X, Y), X is known as extent and Y is known as intent.

Suppose (X_1, Y_1) and (X_2, Y_2) are two concepts of a context, and if $X_1 \subseteq X_2$ (or $Y_2 \subseteq Y_1$), then (X_1, Y_1) will be known as subconcept of (X_2, Y_2) and (X_2, Y_2) will be known as superconcept of (X_1, Y_1). This relation is shown as $(X_1, Y_1) \leq (X_2, Y_2)$ in which $\leq$ is called hierarchical order of the content. The set of all triple concepts of (T, O, I) will be arranged in this way and form a concept lattice. In a dataset, if we define the transactions identifiers as T set, and the features set as O set, we define a procedure to extract the content of datasets using concept lattice.

III. PROPOSED METHOD

In this section CFISDS will be presented for sampling over stream data. Some characteristics of this algorithm are as follows:

- Using a data structure similar to concept lattice to keeping initial transaction in sampling space.
- Insert transactions in concept lattice in batch manner.
- Updating support value and deletion of recently infrequent nodes from concept lattice in a batch manner.
- Using decay mechanism for processing transactional data streams in Landmark window model.
- Using indexing table of concept lattice for fast search in lattice structure
- Sample extraction on demand of user

In the following of describing CFISDS algorithm, first, primary definitions from the used data structure will be explained. Then, the procedure of creating the primary data structure will be discussed. After demonstrating the inserting, updating and incremental deletion of concept lattice, finally the extraction of samples will be explained.

A. Initial Data Structure

The data structure used in CFISDS Algorithm contains two parts:

- A concept lattice which primarily created by an extended Charm-l and then update by an incremental algorithm.
- A sorted array of items with set of pointers assigned to each item. This pointer set, points to some nodes of concept lattice that contain current index table entry.

Fig. 1 shows an example of initiated data structure. In this figure, in order to clarify the figure, just the pointers of item *B* are shown.

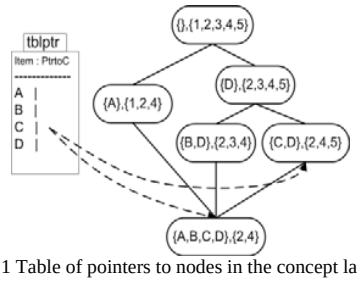


Fig. 1 Table of pointers to nodes in the concept lattice

There are some definitions for a node such as X In a concept lattice, such as ancestors (*anc (X)*), descendents (*desc (X)*), parent (parent (*x*)) and child (*child (x)*). In the following I_X denote object-set of node X.

Definition 1: Node X is the ancestor of node Y, if and only if $I_X \subset I_Y$ and $I_X \neq I_Y$.

Definition 2: Node X is the descendant of node Y, if and only if $I_X \supset I_Y$ and $I_X \neq I_Y$.

Definition 3: If for a node X, $X \in anc(Y)$ and $Z \neq X, \nexists Z \in anc(Y): X \in anc(Z)$, node X considered as parent of node Y.

Definition 4: If for a node X, $X \in desc(Y)$ and $Z \neq X, \nexists Z \in desc(Y): X \in desc(Z)$, node X considered as child of node Y. The following relationship is established in Fig. 2.

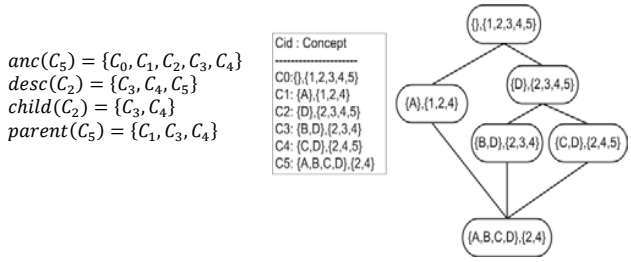


Fig. 2 Example of concept lattice and definitions of ancestor, descendant, parent and child

Data structure used in this algorithm is very important. The data structure, consist of constrained concept lattice and the table of pointers to lattice nodes, causing high-speed search operation, insertion and deletion. There is another algorithm [17] to create concept lattice, but according to the massive number of transactions and objects, the Charm-l has lower memory consumption.

In Charm-l [18], in first step, for each object a set of transactions (TIDset) has created, then using a recursive algorithm called Charm-l-Extend, closed frequent Itemset of objects adds to the lattice. With some modification was made in Charm-l-Extend routine, we can add closed frequent itemsets to lattice and store corresponding transactions in sampling space to build a complete concept lattice. Fig. 3 shows concept lattice created from the original data set.

B. Incremental Updates of Data Structure

In CFISDS algorithm, after the initial data structure created by Charm-l, the incremental update process begins. Each incoming transaction is a closed itemset [19] and according to the decay mechanism [20], the concept insert in the lattice. Decay mechanism helps us to maintain itemsets that have been recently frequent and closed and control the size of the concept lattice. Since the intersection of transactions in the concept lattice structure used to update concept supports, decay mechanism and redundant/non-closed node elimination procedures is significant influence on quality of samples.

Decay mechanism, delete transaction with fewer repetitions by applying incoming time in support value using parameter *d*. Parameter d defined as $(b > 1, h \geq 1)\ d = b^{\frac{-1}{h}}$; where decay-base *b* determines the amount of weight reduction per a decay-unit and decay-base-life *h* is defined by the number of decay-units that makes the current weight be b^{-1}. In addition to the parameter *d*, there is another effective parameter calledS_{sig}. This parameter is defined by the user and used to value initialization, lattice pruning and transaction support update process. In addition, this parameter effect on nodes lifecycle in concept lattice that causes fluctuation in concept lattice size.

In decay mechanism, we use the decay parameter *d* to update the current number of arrived transactions rather than calculating the number of transactions in linear form. $|D|_k$ that represents the current number of transactions calculate using (1).

$$|D|_k = \begin{cases} 1 & k = 1 \\ |D|_{k-1} * d + 1 & k > 1 \end{cases} \quad (1)$$

Due to changes made in data structure and mechanism of reference [26], when transaction T_k (where *k* is the transaction arrival time)insert in concept lattice for the first time, the support value f_k calculate using (2).

$$f_k = |D|_k * S_{sig} * d + 1 \quad (2)$$

In the process of updating the support value for the nodes in concept lattice, the value of this parameter calculate according to the two last incoming transactions related to corresponding node. In (2), *p* is time of transaction arrived before the transaction T_k and update support value of node. Also *k*, shows the current transaction time.

$$f_k = f_p * d^{(k-p)} + 1 \quad (3)$$

In this algorithm, removal process includes 3 mechanisms. In first mechanism, we check the nodes of concept lattice after batch insertion and remove non-frequent nodes according to decay mechanism support value. Elimination of the nodes from concept lattice reduces the support value of transactions related to nodes, and gradually removes linked nodes from the sample space. The second mechanism occurs in the user's request, selects the frequent nodes and eliminates non-frequent of them regard to relative minimum support. By removing nodes, the transactions associated with them are removed from the sample space. The third mechanism also occurs in the user demand; leaves the final samples in sampling space by selecting more related transactions to lattice nodes. More detailed explanation will be given in next session. Fig. 3 depict general procedure pseudo code of the CFISDS algorithm.

```
Input: D- Transactional data stream ,
        S_min– Minimum support threshold,
        batch_size – Size of batch execution,
        d – Decay rate,
        S_sig – Significant support threshold,
        R – Sample size
              Output: Sample – Selected transaction

CFISDS_ALGORITHM (D, S_min, batch_size, d, S_sig, R)
   Dr = first R transaction;
   [Concept_lattice, tblptr]= Modified_Charm_L (Dr, S_min);
   Initiate support for Concept lattice's nodes regard to decay
mechanism;
   D = D-Dr;
   For each block B from D with size batch_size
      search_lattice_and_insert (&Concept_lattice, tblptr, B);
      update_lattice (&Concept_lattice, tblptr, d, S_sig,|D|_k);
   Next block
   Recent_Frequent_node_Selection (&Concept_lattice,
S_min,|D|_k);
   Sample = Select_more_relevant_transaction
(Concept_lattice, R);
```

Fig. 3 pseudo code of CFISDS Algorithm

1) Transaction Processing

Since in our algorithm each incoming transaction t assumed as a closed itemset and will be processed, If this transaction exists as a node in concept lattice, CFISDS updates support value using (3). But in the transaction absence in concept lattice, CFISDS search the lattice structure and insert the transaction in appropriate location. Initial support value of such transactions calculated using (2) according to decay mechanism.

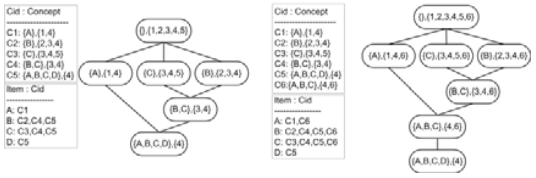
One of the main advantages of the CFISDS algorithm is the index tables of concept lattice. We use Items as entry of index table to find all nodes related to transaction itemset. Equation (4) shows how to obtain these nodes.

$$Condidate\ Nodes\ for\ Transaction\ T = \cup_{i \in T} tblptr(i) \quad (4)$$

It's Necessary to obtain all nodes of concept lattice linked to transaction items. With these nodes we can calculate child and parent relation between nodes. The following example shows importance of using the pointer table.

Suppose we have a concept lattice like Fig. 4.a and transaction {A, B, C} has been arrived. If we use our approach and use nodes set $\{C_1, C_2, C_3, C_4, C_5\}$ to search, the lattice would be like 5.3 and the lattice structure transform like Fig. 4.b as a complete concept lattice structure.

By increasing the number of nodes in concept lattice, the number pointers and the complexity of search procedure increases subsequently. To solve this problem, we eliminate nodes that contain non-used in transaction from candidate node list. Using this procedure, the number of pointers to lattice substantially reduced.



(a) (b)

Fig. 4 Effect of presented method in integrity of concept lattice linkages

In batch search and insertion operation of transactions, in the first step we prepare the list of candidate parents and childs for current node, then reduce this candidate nodes by eliminating nodes contain items that non-used in transaction. At the next step we utilize subset/superset checking procedure to eliminate remained redundant nodes and finally immediate childs and parents remain to make complete links in concept lattice structure for new node.

The growth in concept lattice size cause to increase nodes number in candidate list as child/parent of arrival transaction and increase in search and insert runtime procedure consequently. After an incoming transaction inserted in concept lattice as a new node, the transaction identifier adds to intent part of ancestor nodes. Hence the transaction *t* considered as support of these nodes. We use the batch processing to speed-up of algorithms and then add the informed content to concept lattice.

2) Update and Delete Process

As mentioned in previous section, after batch processing of incoming transactions, the algorithm updates the support value of nodes in concept lattice. During this procedure, we process all nodes in lattice and calculate new support value of updated nodes using (3). After this calculation, we must delete the node if the expression $f_k/|D|_k$ be lower than S_{sig}, otherwise we update f_k to new value according to two last incoming transaction time. In addition, the CFISDS algorithm eliminates the childs of deleted nodes from the lattice. This operation reduces the support value of related transactions in sampling space.

C. Transaction Extraction on User Demand

One of the important features of CFISDS is not only use batch processing in node elimination step of incremental section, but also it can extract recently frequent nodes in any time. If *p* and *k* be two last transactions *ID* in intent of node *e*, where $p < k$, the relevant support value of node *e* calculate as $\left(f_p * d * (k-p)\right)/|D|_k$ according decay mechanism. If this value was lower than minimum support value S_{min}, CFISDS eliminate this node and reduce the support value of related transaction consequently. By reduction of transaction support value, we can remove these samples from sampling space. Fig. 8 illustrates the pseudo code of this algorithm.

Since we know support value of more relevant transactions in sampling space, we can leave top R transaction in sampling space according to relative support value. Here, R is the size of our sampling space.

IV. Concept Lattice Evaluation

The concept lattice is the result of closed frequent mining task. Since we use minimum support value to remove infrequent nodes from concept lattice, finally we have a constrained concept lattice. We expect the similarity of concept lattice was obtained using a closed frequent itemset mining algorithm over CFISDS samples will be more similar to result of these algorithms over original data compare to non-content or other task based sampling algorithms. Despite we have different similarity measures to compare discrete sets e.g. Symmetric difference [21], there aren't a particular measure to compare two concept lattices. Equation (5) show above measures which use *x,y* as two input discrete sets.

$$\text{Symmetric difference } S_{Xor}(x,y) = 1 - \frac{|x \ominus y|}{|x \cup y|} \tag{5}$$

We represent new form for above equations.

Let $\exists S | x, y \subseteq S$, and for *x,y* as two discrete sets, *Max(x,y)* return the maximum value of two sets *x,y*. Now we define two number $m, n \in S$ such that $Max(x,y) \leq m * n$. Using this definition, discrete set *x* reshape to its binary form like (6).

$$BinMat_x = \left[BMX_{i,j}\right]_{m*n}$$
$$BMX_{i,j} = \begin{cases} 1 & \left((i-1)*n+j\right) \in x \\ 0 & \left((i-1)*n+j\right) \notin x \end{cases} \tag{6}$$

With binary representation of sets *x* and *y*, we define Union and Exclusive-Or Matrices.

$$Mat_{Union}\left(BinMat_x, BinMat_y\right) = \left[MUnion_{i,j}\right]_{m*n} \qquad MUnion_{i,j} =$$
$$\begin{cases} 1 & BinMat_{x_{i,j}} = 1 \text{ OR } BinMat_{y_{i,j}} = 1 \\ 0 & BinMat_{x_{i,j}} = 0 \text{ AND } BinMat_{y_{i,j}} = 0 \end{cases} \tag{7}$$

$$Mat_{Xor}\left(BinMat_x, BinMat_y\right) = \left[MXor_{i,j}\right]_{m*n}$$
$$MXor_{i,j} = \begin{cases} 1 & BinMat_{x_{i,j}} \neq BinMat_{y_{i,j}} \\ 0 & BinMat_{x_{i,j}} = BinMat_{y_{i,j}} \end{cases} \tag{8}$$

According these definitions we define "*Ones*" function on binary matrix of a discrete set like (7).

$$\text{Ones}(BinMat_x) = \sum_{i=1}^{m} \sum_{j=1}^{n} BMX_{i,j} \tag{9}$$

Clearly, "Ones" function enumerates 1 values in a binary

matrix. Suppose we have two concept lattices $\mathcal{B}(G,M,I_1)$ and $\mathcal{B}(G,M,I_2)$ that we name them as $\mathcal{B}_1$ and $\mathcal{B}_2$ respectively. Since the relation of concept lattice is similar to a binary matrix, (5) reform to (10) to compare binary relation matrices of lattice $\mathcal{B}_1$and $\mathcal{B}_2$.

$$S_{sym_{BinMat}}\left(Mat_{\mathcal{B}_1}, Mat_{\mathcal{B}_2}\right) = \frac{\text{Ones}\left(Mat_{Intrsct}\left(Mat_{\mathcal{B}_1}, Mat_{\mathcal{B}_2}\right)\right)}{\text{Ones}\left(Mat_{Union}\left(Mat_{\mathcal{B}_1}, Mat_{\mathcal{B}_2}\right)\right)} \quad (10)$$

As described above, relation matrix of concept lattice is convertible to discrete set and vice versa, therefore the presented approach to compare two concept lattices, has essential features of similarity measures.

V. Experimental results

The Sampling test was carried out on a PC with 2.6 GHz processor, main memory of 2 GB and Windows XP as OS. All algorithms was implemented in Matlab. To accuracy Improvement results for each dataset, the sampling operation runs on 10 shuffles of each dataset. Based on the presented definitions in section 4, we use $S_{sym_{BinMat}}$ to compare the obtained concept lattices.

In order to perform practical experiments four synthetic datasets used in which generated by IBM Quest synthetic dataset generator [22]. As well as these, two datasets BMS-Pos and BMS-Web-View-1 were also used. Table II shows the features of these datasets.

TABLE I
CHARACTERISTICS OF USED DATASETS

Data Set	Avg len of Trans	Max len of Trans	# Trans	# Unique Items
T3I4D100K	3	19	69283	1000
T5I4D100K	5	22	84040	1000
T8I4D100K	8	25	95510	1000
T10I4D100K	10	28	98297	1000
BMS-Web-View-1	2.5	267	95602	497
BMS-POS	6.5	164	515597	1657

The distribution of the range of datasets was selected variously to show its effects on algorithms. Since CFISDS algorithm is a sampling algorithm over data stream, in practical tests we compare this algorithm to Z and DSS algorithms. The size of sampling space was selected 7000, and in DSS R=1000. The specific parameters of CFISDS are as follows: b=2, h=10000, batch-size=1000. Table 3 shows the parameters in CFISDS for each datasets.

TABLE II
OPTIONS CFISDS ALGORITHMS FOR DIFFERENT DATA SETS

Dataset	Minimum Support	Ssig
T3I4D100K	0.005	0.3
T5I4D100K	0.005	0.3
T8I4D100K	0.005	0.3
T10I4D100K	0.005	0.3
BMS-Web-View-1	0.005	0.5
BMS-POS	0.01	0.5

As was expected and are shown in Fig. 5, the runtime of CFISDS algorithm is more than DSS and Z. These differences clearly are seen in dense datasets, namely T10, T8, T5 and BMS-Pos. This is due to expansion of concept lattice in which causes increase in the time of search process and creation and deletion of transactions.

In T3 and BMS-Web-View-1 datasets, the runtime of our algorithm is less than DSS. It's duo to the low time to create initial data structure in CFISDS compare to corresponding time in DSS. The small length of transactions effects on the speed of search procedure and creation and deletion.

Eventually, the proposed algorithm initiate its data structure and insert them in sampling space regard to initial data contents. However, DSS create a histogram and use it for keeping information using statistical features of data.

The obtained samples from sampling algorithms with original data set are given to Charm-l algorithm to extract their concept lattice. Table 4 shows the minimum support for each dataset. The selected minimum supports for Charm-l in all dataset are equal to minimum support in CFISDS. In BMS-Pos dataset, due to high density of data in CFISDS samples, the runtime of Charm-l over samples of 400k and 500k datasets is too high. Due to this, we use 0.025 as minimum support.

TABLE III
SELECTED MINIMUM SUPPORT IN CHARM-L ALGORITHM FOR DATA SETS

Dataset	Minimum Support
T3I4D100K	0.005
T5I4D100K	0.005
T8I4D100K	0.005
T10I4D100K	0.005
BMS-Web-View-1	0.005
BMS-POS	0.025

After Charm-l execution over samples and original data, the size of concept lattices was presented in Fig. 6. As expected, the sizes of obtained concept lattices from CFISDS are larger than those were obtained from other samples and original datasets. On the other hand, CFISDS operate on landmark window model therefore the informed contents of its samples increase with growth size of datasets and, in contrast with other sampling algorithms it's not fixed.

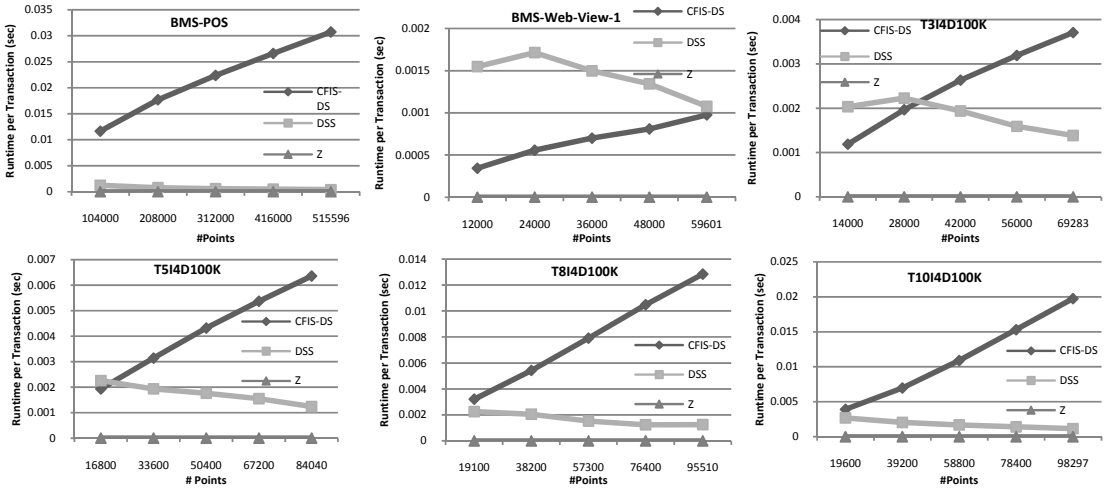


Fig. 5 Runtime per transaction in datasets

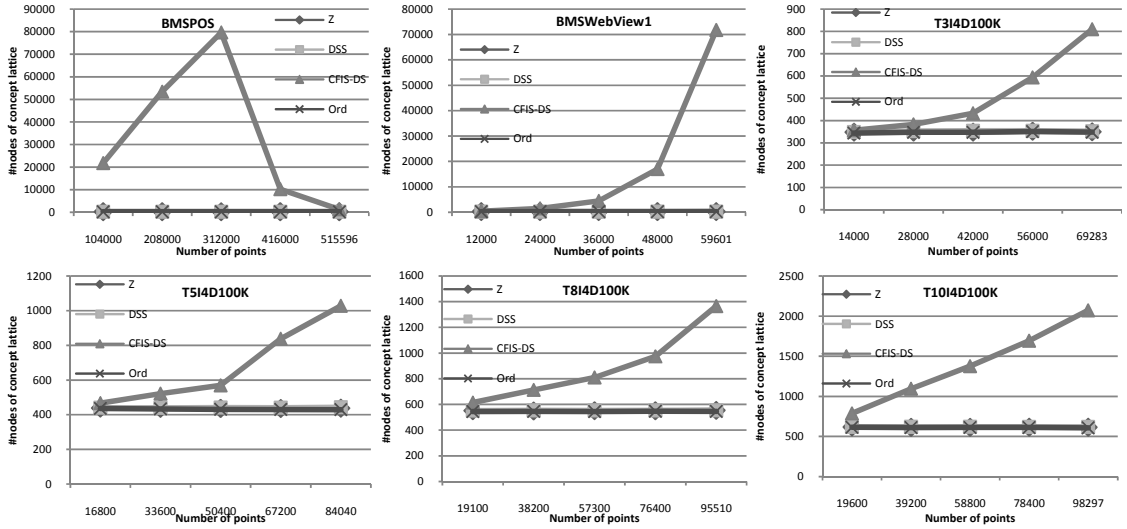


Fig. 6 Size of the concept lattice obtained by execution of extended Charm-l algorithm on the original data and samples

In BMS-Pos dataset in 100k, 200k and 300k dataset size like other datasets, we are seen increase in the size of concept lattice, but due to high amount of minimum support in 400k and 500k , the size of concept lattice are reduced extremely.

The obtained concept lattice from original datasets compare with concept lattice of samples using $S_{sym_{BinMat}}$ measure, are presented in Fig. 7.

As we see in Fig. 7, the value of $S_{sym_{BinMat}}$ measure for concept lattice of CFISDS samples is clearly higher than of those of the other two algorithms. The proposed algorithm could obtain the better result in both synthetic and real datasets. CFISDS algorithm keep more informed samples of transactional data streams using landmark model. In BMS-Pos and BMS-Web-View-1 which considered as sparse dataset, there's high difference between other algorithms in both measures. Although, with increase in average of transactions' length in synthetic datasets the average of $S_{sym_{BinMat}}$ consequently decreases, the presented algorithm has shown a higher efficiency than the other algorithms.

VI. CONCLUSION

In this paper, we describe CFISDS algorithms to extract appropriate and effective samples for closed frequent itemset mining task over data streams. The algorithm, use the concept lattice structure preserves more relevant samples to concept lattice in sampling space. Search, insert, update and delete in batch manner are the most important features of our algorithm in which effect the speed of algorithm. Also due to the algorithm is proposed on landmark model, the obtained contents by our algorithms are more informed compare to other sampling algorithms. Moreover, the CFISDS algorithm can extract the samples which associated with the lattice on demand of user. As disadvantages of this algorithm, we can state several user-defined parameters and weakness to handle massive number of features or transactions.

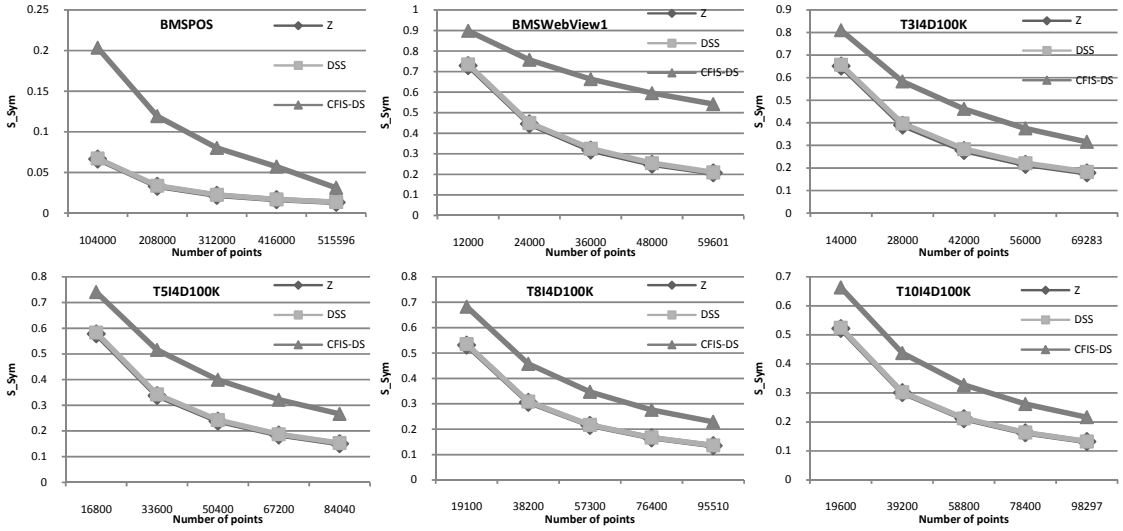


Fig. 7 Comparison of the concept lattices obtained from the original datasets and samples using $S_{Jac_{BinMat}}$ measure

REFERENCES

[1] J. Vitter; "Random sampling with a reservoir", ACM Transactions on Mathematical Software, Vol 11(1), pp.37– 57, 1985 .

[2] P.B.Gibbons, Y. Matias. "New sampling-based summary statistics for improving approximate query answers". In Proceedings of ACM SIGMOD international conference on management of data, 1998, pp.331–342.

[3] C. Estan, G. Varghese. "New directions in traffic measurement and accounting" . In: Proceedings of 1st ACM SIGCOMM workshop on Internet measurement, 2001, pp. 75–80.

[4] G.Manku, R.Motwani. "Approximate frequency counts over data streams". In Proc. 2002 Int. Conf. Very Large Data Bases, 2002, pp. 346-357.

[5] E.D.Demaine, A.L'opez-Ortiz, J.I.Munro. "Frequency estimation of Internet packet streams with limited space". In: Proceedings of 10th European symposium on algorithms, 2002, pp. 348–360.

[6] M. Dash, W. Ng. "Efficient Reservoir Sampling for Transactional Data Streams" . Sixth IEEE International Conference on Data Mining - Workshops, 2006, pp. 662-666 .

[7] W.Ng, M.Dash. "Which Is Better for Frequent Pattern Mining: Approximate Counting or Sampling?". In Proceedings of the 11th international Conference on Data Warehousing and Knowledge Discovery , 2009, p.p. 151-162.

[8] J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD the Internatial Workshop on Data Mining and Knowledge Discovery*, 2000.

[9] J.Wang, J.Han and J.Pei: CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets; In *Proceeding of ACM SIGKDD the InternationalConference on Knowledge Discovery and Data Mining*, 2003

[10] M.J.Zaki and C.Hsiao: Charm: An efficient algorithm for closed itemset mining; In *Proceedings of SDM the SIAM International Conference on Data Mining*, 2002

[11] C. Lucchesse, S. Orlando, and R. Perego. DCI-Closed: A fast and memory efficient algorithm to mine frequent closed itemsets. In B. Goethals, M. J. Zaki, and R. Bayardo, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2004), volume 126 of CEUR Workshop Proceedings, Brighton, UK, 1 November 2004.

[12] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In B. Goethals and M. J. Zaki, editors, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2003), volume 90 of CEUR Workshop Proceedings, Melbourne, Florida, USA, 19 November 2003.

[13] T.Uno , T.Asai , Y.Uchida , H.Arimura . LCM: An efficient algorithm for enumerating frequent closed item sets , In Proceedings of Workshop on Frequent itemset Mining Implementations , 2003 .

[14] B.N. Ranganath, M.N. Murty. "Stream-Close: Fast Mining of Closed Frequent Itemsets in High Speed Data Streams". ICDM Workshops 2008, 2008, pp. 516-525.

[15] H.Li, H.Chen. "Moment+: Mining Closed Frequent Itemsets over Data Stream", ADMA 2008, 2008, pp. 612-619.

[16] Y. Chi, H. Wang, P.S. Yu, R.R. Muntz. "Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window", In Proc. Fourth IEEE Int'l Conf. Data Mining, 2004, pp. 59-66.

[17] V. Choi. (2006, Jun) "Faster Algorithms for Constructing a Concept Galois Lattice." CoRR .vol abs/cs/0602069. Available: http://arxiv.org/abs/cs/0602069 [Jun. 1,2006].

[18] M.J. Zaki. C.-J. Hsiao. "Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure". *IEEE Trans. on Knowl. and Data Eng.* Vol.17, No.4. pp. 462-478. 2005.

[19] P. Valtchev, R. Missaoui, R. Godin. "A framework for incremental generation of closed itemsets". Discrete Applied Mathematics vol.156 (6). pp. 924–949. 2008.

[20] J.H. Chang, W.S. Lee. "Finding recently frequent itemsets adaptively over online transactional data streams". *Inf. Syst.* Vol.31, No.8. pp. 849-869. 2006.

[21] F. Alqadah , R. Bhatnagar. "Similarity Measures in Formal Concept Analysis". ISAIM 2010, Fort Lauderdale, Florida , 2010.

[22] IlliMine. Package for Data Mining in C++ ,[Online] Available: http://illimine.cs.uiuc.edu.