TELKOMNIKA, Vol. 11, No. 8, August 2013, pp. 4780~4787
e-ISSN: 2087-278X

Frequent Itemsets Mining based on Concept Lattice and Sliding Window

Zhang Chang-sheng[1,2,3], Ruan Jing[4], Huang Hai-long*[3], Li long-chang [3], Yang Bing-ru[1,2]
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083, China
[2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, 100083, China
[3]College of Physics & Electronic Information Engineering, Wenzhou University, Zhejiang, 325035, China
[4]Wenzhou Vocational & Technical College, Zhejiang, 325035, China
Corresponding author, e-mail: jsj_zcs@126.com, ruanjing1979@126.com, 156732998@qq.com, jsj_llc@wzu.edu.cn, bryang_kd@126.com

Abstract

In this paper, a frequent itemsets mining algorithm of data stream based on concept lattice and sliding window is presented. This algorithm mines frequent concepts for new inflowing basic window in batches in a sliding window and generates concept lattice Hasse diagram. With introduction into small support degree ζ and error factor ε to do the pruning operations for non-frequent concept node, each connection point in the Hasse diagram contains the information of frequent itemsets and support degree. As the generation of Hasse diagram in the new basic windows, we integrate concept lattice vertically with the generated Hasse diagram and sliding window, and ultimately output all frequent itemsets through scanning all the graph nodes of Hasse diagram graph. The experimental results show that the proposed algorithm has a good performance.

Keywords*: data steam, frequent patterns, sliding window, concept lattice*

1. Introduction

With the growing number of data applications, the data mining techniques have attracted widespread attention under environment of data stream. However, data stream has characteristics of continuous, unlimited, real-time, and un-prediction, and the traditional frequent pattern mining algorithms are difficult to deal with such data stream. Therefore, frequent pattern mining under the environment of data stream has become a challenging research direction. Some relevant algorithms are put forward successively, such as Sticky Sampling and Lossy Counting in the literature [1] gave an effective algorithm for solving a single frequent pattern by introducing the error factor ε, this algorithm can obtain the entire frequent itemsets of data stream by scanning data. The FP-Stream algorithm was presented in the literature [2]. This algorithm uses the Pattern-Tree structure with a similar FP-Tree prefix tree to store the potential frequent pattern information for the time window in the past, and solves the time sensitivities problem of historical data by the introduction of tilted time window techniques. Chang in the literature [3] proposed a mining frequent itemsets algorithm SW by making use of sliding window. The concept of frequent closed patterns was proposed by Pasquier in the literature [4], it is only to determine the accurate support rate of all frequent patterns, and the size is of smaller magnitudes of order than the frequent pattern set. Chi in the literature [5] provided closed frequent itemsets mining algorithm based on sliding window.

Since the theory of formal concept analysis was put forward according to the philosophy of "concept" thinking by Germany professor Wille since 1982 [6], its core data structure of the formal concept analysis theory is-concept lattice. Relevant researches have obtained rapid development, several research directions on the combination of concept lattice and data mining has gradually become a hot topic. Because the essence of concept lattice is a concept of hierarchical structure induced by a binary relation, it is very suitable for data analysis and rule extraction. According to the generated process of concept lattice by number of data records in data sets, its essence is a clustering process for concepts. And it is a very useful formal analysis tool. The concept lattice embodies the unification for the concepts of connotation and

Received March 8, 2013; Revised May 25, 2013; Accepted June 3, 2013

denotation, it not only reflects the association between objects and attributes, but also contains the concept relationships between generalization instantiation. Therefore, it is very suitable to find the potential concept and knowledge with combinations with data mining applications.

At present, group of scholars at home and abroad have already studied the combined application of concept analysis and data mining, the main research directions focus on some quick concept lattice algorithms [7, 8], the association rule algorithms based on concept lattice [9, 10], the classification algorithms based on concept lattice [11], the relationships between the concept lattices and rough sets [12, 13], and other aspects.

There are few works about mining frequent itemsets in data stream based on concept lattice by access to domestic and foreign literature. Learn from previous researches on data stream mining and concept lattice, in this paper, we present a new frequent pattern mining algorithm DSCL in data stream based on sliding window and concept lattice. The proposed algorithm makes use of a core data structure concept lattice Hasse diagram in the formal concept analysis theory to store some potential frequent concept in the sliding window, as the sliding window updating and maintaining this structure in real time constantly, we can output the frequent itemsets.

2. Related Definitions

2.1. Sliding Window

Definition 1. Suppose $I=\{i_1,i_2,\ldots,i_n\}$ be a set of all data items, itemset X is a subset of complete data items I sub-sets ($X \subseteq I$), some items containing k items are k-itemsets. Transaction T is an itemset, data stream can be seen as a continuous arriving transaction sequence $DS = \{T_1, T_2, ..., T_N\}$. Let T_1 be the transactions of earliest arrival time in data stream, T_N be the latest arrival transactions in the data stream.

Definition 2. Let w represent the fixed size of basic sliding window, that is, there are only w recent transactions in the basic sliding window. Data stream DS can be segmented according to the number W, each w transactions correspond to as a sub-sequence data stream, that the corresponding size (or width) of the basic sliding window is **w**. The current basic sliding window is represented as: $sw_i=\{T'_1,T'_2,\ldots,T'_w\}$, where *sw* indicates the basic sliding window, i is the current window number of the window (i.e., the i-th basic window). The sliding window SW consists of a continuous series of basic window sw_i, which denotes as $<sw_1, sw_2,\ldots, sw_k>$, sliding window contains the numbers of the window is the size of the sliding window, denoted by | SW | = k.

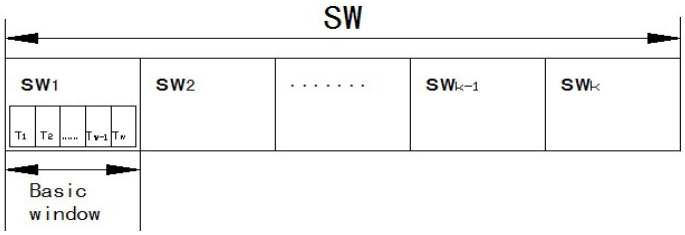

Figure 1. Sliding Window

Definition 3. Given a minimum support threshold ζ and error factor ε, suppose w denote the width of the sliding window SW, that SW contains number of transactions. $f_{sw}(A)$ denotes the support count of pattern A in the sliding window SW. For pattern A, if there is $f_{sw}(A) \geq (\zeta - \varepsilon)w$, then A is a frequent pattern in the sliding window SW; if there is $f_{sw}(A) > \varepsilon w$, then A is a critical frequent pattern in the sliding window SW; if there is $f_{sw}(A) \leq \varepsilon w$, then A is a non-frequent pattern in the sliding window SW; for the basic window *sw* also has the same definition as above.

2.2. Concept Lattice

In the formal concept analysis, it can be understood as follows, the extension of a concept C is a set of all objects that belong to the C, the common feature or attribute set of all

these objects is called the connotation of the concept C, and each concept in concept lattice is a set of objects of the greatest common properties. All concepts together with the relationships of generalization/instantiation form the concept lattice. The concept lattice is the core data structure of formal concept analysis theory, the corresponding Hasse diagram realize data visualization.

Definition 4: A formal context (Context) is a triple K =(G, M, *I*), where G is a set of objects, M is a set of attributes, *I* is a binary relation between the G and M, i.e. $I \subseteq G \times M$, g*I*m denotes it exists a relationship I.

Definition 5: In the formal context K, a binary groups (A, B) from G × M exists the following two properties:

(1) B =f(A), where f(A) ={m:(m∈M)∧(∀ g∈A⊆G, g*I*m)};

(2) A =g(B), where g(B)={g :(g∈G)∧(∀ m∈B⊆M, g*I*m)}.

In the formal context K, (A, B) is called as a concept, where B is referred to as intent of the concept (Intent), and A is called the extension of the concept (Extent).

Definition 6: A partial order relation between the concept nodes is established. Given $C_1=(A_1, B_1)$ and $C_2=(A_2, B_2)$, then $C_1>C_2 \Leftrightarrow B_1 \subset B_2 \Leftrightarrow A_1 \supset A_2$, the leading order means C1 is the parent node of C2 or the generalization. If concepts $C_1=(A_1, B_1)$ and $C_2=(A_2, B_2)$ satisfy $A_2 \subset A_1$, and there does not exist the concepts (A,B) such that $A_2 \subset A \subset A_1$; then C1 is called the direct super-concept of C2, C2 is a direct sub concept of C1, referred to as $(A_1, B_1)>(A_2, B_2)$. The linear diagram of concept lattice is generated based on the partial order relation, that is Hasse diagram.

Definition 7: Given a set of objects G, and a set of attributes M, a binary relation is $I \subseteq G \times M$ (here (g, m) ∈ I is called as "object g has attribute m").

Definition 8: For the concept C(X, Y), C' (|X|, Y) is quantized by the concept C, and C is the real concept of C'. |X| is the cardinality of the epitaxial X, the posed lattice is quantified by quantitative concept, which is quantitative concept lattice.

Definition 9: Given concept C(A,B), B∈M; given a threshold minsup∈[0, 1], the support degree of attribute set B in forms background K is $\sup(B)=|(g(B)|/|G|$ (where g(b)={g∈G| ∀ m∈ B: (g,m)∈*I*}), if sup(B)≥minsup, then B is called the frequent attribute set (content), and C is a frequent concept. That if the connotation of concept is frequent, then the connotation is called as the concept of frequent connotation, which is also called as frequent concept, $\sup(C)=|(g(B)|/|G|=|A|/|G| \geq \min\sup$ where | A | is the epitaxial base of concept C, |G| is the total number of transactions in the database D.

3. DSCL Algorithm

3.1. The Descriptions of Algorithm

Algorithm DSCL divides data stream into some data blocks of equal length, each data block is as a basic window *sw*, each window contains the number of transactions w, and k consecutive basic windows are composed of a sliding window SW. The algorithm generates a Hasse diagram after the first window entering into the sliding window, and after each new window entering into the sliding window the new Hasse diagrams are generated, and they merge vertically and reconstruct with previous Hasse diagrams in real-time. After completing the establishment and consolidation of Hasse diagram for basic window and generate new Hasse diagram that are reference to error factor ε and the minimum support threshold ζ to filter out non-frequent concept (pruning) and to reconstruct the sides of Hasse diagram. The Algorithm just need to handle and store the critical frequent concept set of each basic window, it does not contain non-frequent concept node. Therefore, this algorithm greatly reduces the processing time and the amount of data storage. It consists of four main parts: the initialization of window, the stage of window sliding, the pruning of window Hasse diagram and the merger of window Hasse diagram.

3.2. Pseudo Codes

Algorithm 1: DSCL

Input: The basic windows sw_1, sw_2, sw_3, …sw_w…sw_n of data stream, the minimum support degree ζ, the error factor ε=xζ, the size of sliding window k, the size of basic window size w.

Output: the frequent concept lattice Hasse diagram.

Step 1:

i=1; //i is the identification of basic sw_i

n=0; //n is the count of the current sliding window SW containing basic sw_i

ε=xζ ;// error factor, x is adjustable

minsupw= ε w; // The critical support count of basic window

minsupk= ε k w; // The critical support count of sliding window

minsupn= ε (n+1) w; // The critical support count of sliding window for n +1 basic windows

Hall-i= $\varnothing$; //Current Hasse diagram Hall-i is empty

Flag=false; // Whether the sliding window is full window

Step 2: Input basic sliding window sw_i, use 0,1 to indicate whether the transaction owns the attribute and generate a formal context matrix.

Step 3: sw_i=>Hasse diagram H_i ; // use the improved algorithm to generate concept lattice in batch swi of Hasse diagram H_i;

DeleteConceptLattice (H_i, minsupw); // Delete no-frequent concepts

Step 4: IF n>=k

Flag=true; // Flag=true is full window

END IF

Step 5: IF Flag = = false // Flag=true is full window

IF $H_{all\text{-}i\text{-}1} \neq \phi$

$H_{al1\text{-}i\text{-}1}$+$H_i$ => $H_{all\text{-}i}$;// call the improved vertical integration of concept lattice algorithm to combine and generate the Hasse diagram $H_{all\text{-}i}$

Let Minsupn= ε ζ (n+1) w; // the support count of the sliding window for current n+1 basic windows

DeleteConceptLattice($H_{all\text{-}i}$, minsupn); // Delete no-frequent concepts

n=n+1; // Record the number of basic windows entering into the sliding window

ELSE IF $H_{all\text{-}i\text{-}1}$= = ϕ

Let H_i=> $H_{all\text{-}i}$; //Initialize $H_{all\text{-}i}$ is H_i

END IF

ELSE

delete $H_{i\text{-}k}$ From $H_{all\text{-}i\text{-}1}$; // Delete Hasse diagram $H_{i\text{-}k}$ of basic window $sw_{i\text{-}k}$ in $H_{all\text{-}i\text{-}1}$

$H_{al1\text{-}i\text{-}1}$+$H_i$ => $H_{all\text{-}i}$; // call the vertical integration of improved concept lattice algorithm to generate the Hasse diagram $H_{all\text{-}i}$ of combining H_i and $H_{all\text{-}i\text{-}1}$

DeleteConceptLattice(H_i, Minsupk); // To delete no-frequent concepts

END IF

Step 6: i=i+1, turn to Step 2. // Get the next basic window sw_{i+1}

Algorithm 2: DeleteConceptLattice (H ,Minsup_Num)

Input: Hasse diagram H of one concept lattice Minimum support count Minsup_Num

Output: Hasse diagram H'.

```
Begin
    For each C ∈ H //C is a node of H
    If sup(C) ≤ Minsup_Num
    Delete C from H; // Delete no-frequent concepts C from H, do the pruning operation
    End For
    For each C ∈ H
    Re_Add edge C =>H; // Reconstruct side for each concept node
  End For
END
```

3.3. Instance of Verification

The preceding 16 transactions in data stream DS determine the formal context K as shown in Table 1, w=4, k = 3, ζ = 30%, ε=0.33ζ .

Table 1. Formal Context

ID	a	b	c	d	e	f	g	ID	a	b	c	d	e	f	g
1	1	0	1	1	0	1	0	9	1	1	0	1	0	1	0
2	1	1	1	1	0	1	0	10	1	1	1	1	0	0	1
3	1	0	1	0	0	0	1	11	1	1	1	0	0	0	0
4	1	1	0	0	0	0	1	12	1	0	0	0	0	0	1
5	1	1	0	0	0	0	1	13	1	1	0	0	0	1	1
6	1	1	1	0	0	0	1	14	1	1	1	0	0	0	0
7	1	1	0	1	0	1	0	15	1	1	1	1	0	0	1
8	1	0	1	1	1	0	0	16	1	0	1	1	0	1	0

Step 1:// Initialize parameters

i=1; // i is the identification of basic sw_i

n=0; // The count of basic window containing some basic windows

ε=0.33ζ =0.33*0.3=0.1;// error factor

Minsupw= ε w=0.4; // The critical support count of basic window

Minsupk= ε k w=1.2; // The critical support count of sliding window

Minsupn= ε (n+1) w=0.4;// The critical support count of only one basic window

$H_{all-0}=\phi$; //current Hasse diagram H_{all} is empty

Flag=false;//whether the sliding window is full

Step 2: The sliding window $SW_1$is is empty, which is a non-full window. The inflowing data of the basic window SW_1 is T_1, T_2, T_3, T_4. We use 0,1 to indicate that the transaction whether has this attribute and generate a formal context matrix, such as the first to fourth row in Table 1.

Step 3: The improved batch algorithm let sw1 generate $Hasse_1$ diagram denoted as H_1 , due to Minsupw=0.4, which is less than the minimum support count 1 of the concept nodes in H_1, and we do not need to call Delete Concept Lattice (H_1,Minsupw), the result of this step is shown in Figure 2.

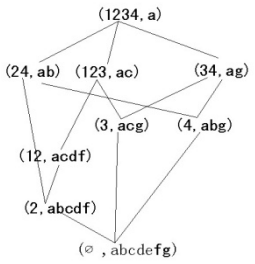

Figure 2. H_1 Diagram

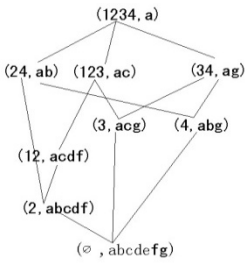

Figure 3. H_{all-1} Diagram

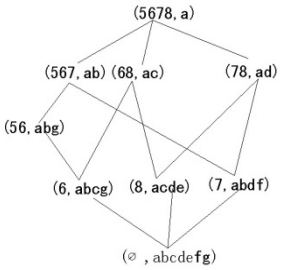

Figure 4. H_2 Diagram

Step 4: Due to n=0; thus n>=k does not satisfy, Flag=false, the current sliding window is not a full window.

Step 5: Because Flag = false and H_{all-0}= =ϕ, thus H_1=> H_{all-1}, the current concept lattice H_{all-1} of sliding window, which shown in Figure 3.

Step 6: i= i +1, turn to Step 2, as described above steps, continue to complete the input of sw_2 and sw_3 the results are as follows:

The inflowing data of the basic window sw_2 is T_5、T_6、T_7、T_8, and the inflowing data of the basic window sw_3 is T_9、T_{10}、T_{11}、T_{12}, such as the fifth to eighth row in Table 1, the ninth to twelfth row in Table 1.

The generated $Hasse_2$ diagram of sw_2 is denoted by H_2, as shown in Figure 4; combine H_2 and H_{all-1} to generate the H_{all-2} shown in Figure 5, because Minsupn= ε (1+1) w=0.8 is less than the minimum support count of concept lattice in H_{all-2} diagram. Thus it does not call Delete Concept Lattice (H_{all-2}, Minsupn);

The generated Hasse$_3$ diagram of sw_3 is denoted by H_3, as shown in Figure 6; combine H_3 and H_{all-2} to generate the H_{all-3} shown in Figure 7, because Minsupn= ε (2+1) w=1.2 is less than the minimum support count of concept lattice in H_{all-3} diagram. Thus it does not call DeleteConceptLattice (H_{all-3}, Minsupn), and delete the concept nodes of non-frequent concept lattice, and generate H'_{all-3} in Figure 8.

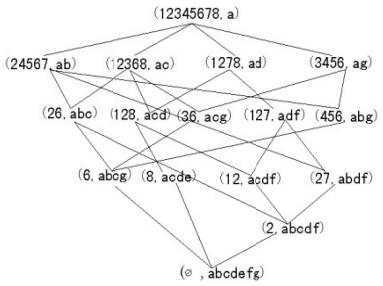

Figure 5. H_{all-2} Diagram

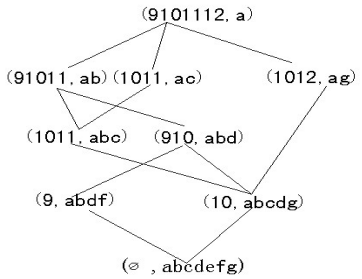

Figure 6. H_3 Diagram

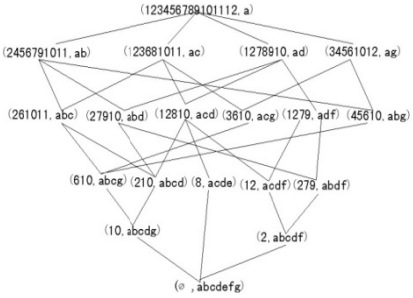

Figure 7. H_{all-3} Diagram

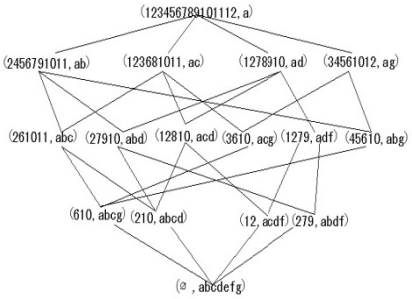

Figure 8.H'_{all-3} Diagram

Up to this step so far, the number of size of sliding window containing the basic windows, then the sliding window is full.

Step 7: i= i +1, Step 2-Step 6 are repeated, continue to complete the input and processing of sw_4, the results are as follows: the inflowing data of basic window sw_4 is T_{13}、T_{14}、T_{15}、T_{16} , generate a formal context matrix in the in the 13th to the 16th row shown in Table 1.

The generated Hasse H_4 diagram denoted by H_4, as shown in Figure 9; and execute to Step 4, since n=k=3, so Flag=true; and then perform Step 5, delete firstly the concept nodes in H_1 from H_{all-3} and generating H'_{all-3}, as shown in Figure 10. Then, call the improved vertically merging algorithm of concept lattice and combine H_4 and H'_{all-3} to generated Hasse diagram H_{all-4}, as shown in Figure 11; and call DeleteConceptLattice (H_i, Minsupk) to delete non-frequent concept node in H_{all-4} and generate H'_{all-4} , the final result show in Figure 12.

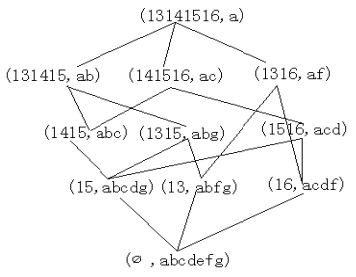

Figure 9. H_4 Diagram

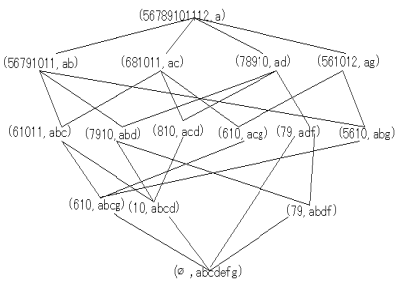

Figure 10. H'_{all-3}

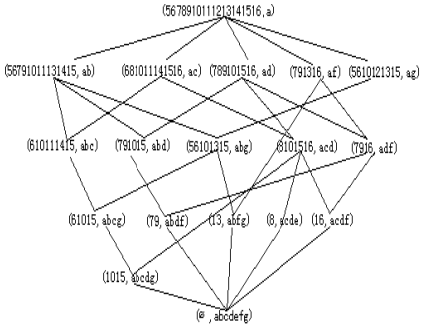

Figure 11. H_{all-4}

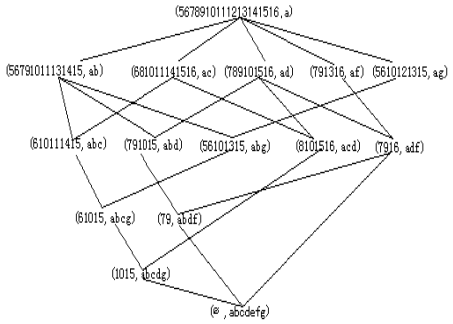

Figure 12. H'_{all-4}

Step 8: i= i +1, repeat Step 2-6, and continue to complete the input and processing of sw_5, sw_6,… sw_n.

4. Experimental Analysis

Because FP-stream algorithm has good time efficiency, we select the FP-stream as a comparison algorithm. The experimental environment is a Pc machine of Windows Server 2003 operating system, I7 2.0G 64-bit quad-core eight lines into the processor, 8G memory, and the program runs on the JAVA SDK1.4.2 environment. The experiments [14] employ the data set T10I6D100K generated by IBM synthetic data generator, where the total number of transactions are100,000, the average transaction length is 10, the average length of the potential frequent itemsets is 6, this data set contains different item number 1000. The experiments make a comparison with Algorithms FP-stream and DSCL, not considering the read time of data. Figure 13 presents a comparison of mining time consumption as the number of transactions of the sliding window change under the premise of the conditions that the window size w changes, the minimum support ζ is 0.003, the error factor is 0.1 ζ ; Figure 14 shows a comparison of memory consumption as the minimum support ζ changes between Algorithms FP-stream and DSCL under the premise of the conditions that the window size W = 1000, the error factor takes 0.1 ζ.

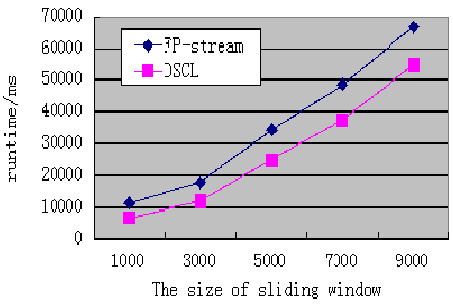

Figure 13. Comparisons of Running Time of Algorithms FP_Stream and DSCL

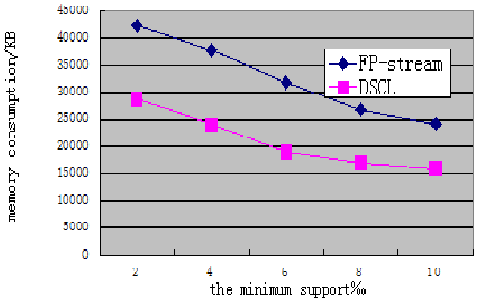

Figure 14. Comparisons of Memory Consumption of Algorithms FP_Stream and DSCL

The experiments have demonstrated the solving efficiency is similar between the DSCL algorithm and the FP-stream algorithm based on sliding time window and concept lattice. A good performance of this algorithm is to use Hasse diagram of the concept lattice to indicate frequent concept sets, the next is to compress the size of the concept lattice efficiently under the case of ensuring the integrity of information followed by the oriented concept lattice pruning strategy, greatly reduces the search range of the algorithm and improve the space-time complexities of the algorithm.

5. Conclusion

In this paper, a new frequent pattern mining algorithm DSCL in data stream based on sliding window and concept lattice, due to it employs a core data structure concept lattice Hasse diagram to store the potential frequent concept of the sliding window in the formal concept analysis theory, and use pruning strategy aiming for concept lattice in the generated process of basic window and sliding window, delete the non-frequent concept in the concept lattice. Because the scale after pruning concept lattice is much less than the corresponding scale of concept lattice, thus it greatly reduce the search scope of solving the frequent itemsets, overcome the drawback of previous construction algorithm on the concept lattice, and improve the space-time performance of solving the frequent itemsets based on concept lattice. The experimental results show that DSCL algorithm has a better time and space complexity. In this paper, we make an experimental under the fixed attributes of experimental data set. Our future research work is to the construction of concept lattice based on sliding window under the case that attributes increase or decrease around the concept solved oriented data flow, and to carry out the algorithm on merging vertically and horizontally, and to mine the frequent patterns and characteristics rules in the case of ensuring the performance of space and time complexities.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No.60875029, No.61175048); The Project of Beijing Key Laboratory of Knowledge Engineering for Materials Science(No.Z121101002812005)

References

[1] Manku G S, Motwani R. *Approximate frequency counts over data streams*. Proceedings of the 28th International Conference on Very Large Data Bases (VLDB). 2002; 346-357.
[2] Giannella C, Han J, Pei J, et al. Mining frequent patterns in data streams at multiple time granularities. Data Mining Next Generation Challenges and Future Directions. Massachusetts: MIT Press. 2004:191-212.
[3] Chang JH, Lee WS. A sliding window method for finding recently frequents itemsets over online data streams. *Journal of Information Science and Engineering*. 2004; 20(4): 753-762.
[4] N Pasquier, Y Bastide, R Taouil, et al. *Discovering frequent closed itemsets for association rules*. Proceeding of the 7th International Conference on Database Theory (DBT). Berlin: Springer-Verlag. 1999; 398-416.
[5] Chi Y, Wang H, Yu PS, Muntz RR. Catch the Moment: Maintaining Closed Frequent Itemsets over a Stream Sliding Window. *Journal of Knowledge and Information Systems*. 2006; 10(5): 265-294.
[6] Wille R. Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rivaled. Ordered Sets, Dordrecht: Riesel. 1982: 445-470.
[7] Zhi Hui-lai, Zhi Dong-jie, Liu Zong-tian. Theory and Algorithm of Concept Lattice Union. *Acta Electronica Sinica*. 2010; 38(2): 455-459.
[8] Gu Chun-sheng. Fully Homomorphic Encryption, Approximate Lattice Problem and LWE. *International Journal of Cloud Computing and Services Science*. 2013; 2(1): 1-15.
[9] Wang Hui, Wang Jing. Non-redundant association rules extraction of frequent concept lattice based on FP-tree. *Computer Engineering and Applications*. 2012; 48(15): 12-14.
[10] Yang Kai, Jin Yong-long, He Zhi-jun, Ma Yuan. An Approach for Briefest Rules Extraction Based On Compact Dependencies. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(2): 941-947.
[11] Zhao Xu-jun, Zhang Ji-fu, Ma Yang, Cai Jiang-hui. Novel Classification Rule Acquisition Algorithm. *Journal of Chinese Computer System*s. 2012; 33(5): 1126 -1130.
[12] Lv Yue-jin, Liu Hong-mei. Improved algorithm for attribute reduction on concept lattice. *Computer Engineering and Applications*. 2011; 47(8): 146-148.
[13] Wang Dong-yan, Huang Ying-hui, Li Guan-yu. Lattice-tree transforming method of generating rough ontology. *Computer Engineering and Applications*. 2012; 48(5): 44-46.
[14] www.almaden.ibm.com/cs/quest/syndatd.html.