An Efficient Method for Mining Temporal Emerging Itemsets in Data Streams

Vincent S. Tseng
Dept. Computer Science and Information Engineering
National Cheng Kung University, Taiwan, ROC
tsengsm@mail.ncku.edu.tw

Chun-Jung Chu
Dept. of Computer Science
National Chiao Tung University, Taiwan, ROC
cjchu@cis.nctu.edu.tw

Tyne Liang
Dept. of Computer Science
National Chiao Tung University, Taiwan, ROC
tliang@cis.nctu.edu.tw

ABSTRACT

In this paper, we propose a new method, namely EFI (Emerging Frequent Itemset)-Mine, for mining temporal emerging frequent itemsets from data streams efficiently and effectively. Discovery of emerging frequent itemsets is an important process for mining interesting patterns from data streams. The novel contribution of EFI-Mine is that it can effectively identify the potential emerging itemsets such that the execution time can be reduced substantially in mining all frequent itemsets in data streams. This meets the critical requirements of time and space efficiency for mining data streams. The experimental results show that EFI-Mine can find the emerging frequent itemsets with high precision under different experimental conditions and it performs scalable in terms of execution time.

1: INTRODUCTIONS

Association rules mining [1, 2] is a well studied topic in data mining field, and an important extended research issue is the discovery of temporal association patterns in data streams due to the emerging applications. Although a number of methods have been designed for mining traditional databases, they cannot perform well for mining temporal patterns in data streams because of the high complexity.

Without loss of generality, consider a typical market-basket application as illustrated in [3] has been considered. The transaction flow in such an application is shown in Figure 1 where items *a* to *i* stand for items purchased by customers.

In Figure 1, for example, the third customer bought item c during time t=[0,1), items c, e and g during t=[2, 3), and item g during t=[4, 5). It can be seen that in such a data stream environment it is intrinsically difficult to conduct the frequent pattern identification due to the limited time and space constraints. Furthermore, it wastes too much time finding frequent itemsets in different window times. Therefore, we develop a new scheme to find potential emerging frequent itemsets before next window times.

Dong and Li [4] define an emerging pattern as an itemset the support of which increases significantly

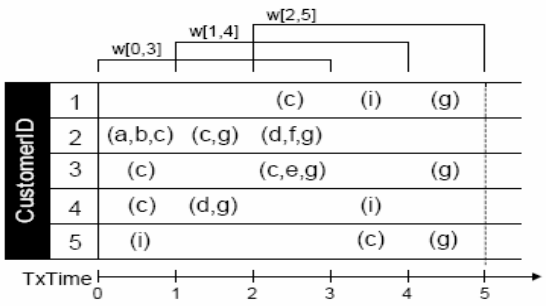

Figure 1. An example of online transaction flows.

between two databases. We view emerging frequent itemsets as a special case of the emerging patterns described by Dong and Li. An *Emerging Frequent Itemset (EFI)* can be considered as an itemset that is infrequent (i.e., small) in the current database and gets increased for its support so that it will eventually become frequent (i.e., large) in the new database temporally added with new data transactions. For example, in the market basket domain, we may assume an interval as the time between wholesale purchases. Recognizing the set of items that will emerge or become frequent in the next time period with windows size may allow the storekeeper to order these emerging items much earlier than usual. Thus, the storekeeper will know what kinds of items will be popular in the next time period and avoid losing the income that their sales could have generated. Although some related issues like mining emerging frequent itemsets [5] and incremental frequent itemsets [7] have been studied, they have been focused on traditional databases and are not suited for data streams.

In this paper, we explore the issue of efficiently mining emerging frequent itemsets in temporal databases like data streams [8]. We propose an algorithm named *EFI-Mine* that can discover emerging frequent itemsets from data streams efficiently and effectively. The novel contribution of *EFI-Mine* is that it can effectively identify the potential emerging frequent itemsets in data streams so that the execution time for mining frequent itemsets can be substantially reduced. Hence, the process of discovering all frequent itemsets under all time windows of data streams can be achieved efficiently with

limited memory space. This meets the critical requirements of time and space efficiency for mining data streams. Through experimental evaluation, *EFI-Mine* is shown to deliver high precision in finding the emerging frequent itemsets and high scalability in terms of execution time.

The rest of this paper is organized as follows: Section 2 gives the definition for the targeted problem. Section 3 describes the proposed approach, *EFI-Mine*, for finding the emerging frequent itemsets. In section 4, we describe the experimental results for evaluating the proposed method. A conclusion is given in Section 5.

2: Problem Definitions

2.1: Support Framework for Mining Temporal Patterns

Consider the transaction flows shown in Figure 1. Given the window size w=3 and the minimum support value as 40%, occurrence frequencies of the inter-transaction itemset {c, g} from time t=1 to t=5 can be obtained as shown in Table 1.

With the sliding window model, the frequent temporal patterns can be discovered for different time windows. The main goal of our research is to discover interesting emerging itemsets under progressive time windows.

2.2: Emerging Frequent Itemsets and Interesting Emerging Itemsets

Table 1: The support values of the inter-transaction itemset {c, g}.

TxTime		Occurrence(s) of {c,g}	Support
t=1	w[0,1]	none	0
t=2	w[0,2]	CustomerID={2, 4}	2/5=0.4
t=3	w[0,3]	CustomerID={2, 3, 4}	3/5=0.6
t=4	w[1,4]	CustomerID={2, 3}	2/5=0.4
t=5	w[2,5]	CustomerID={1, 3, 5}	3/5=0.6

In a database, the frequent itemsets will be changed when new data are added. As time progress, we can see many interesting patterns with regards to the change in status of individual itemsets. An itemset that was infrequent may become frequent (large), while frequent itemsets may become infrequent (small) and an itemset may remain frequent or infrequent. We define infrequent itemsets that are moving toward being frequent as *emerging*. Conversely, frequent itemsets moving toward infrequent are *submerging*. An infrequent (frequent) itemset that becomes large, i.e. with support above (below) minimum support value, is said to have emerged (submerged). The problems we address in this paper are: 1) How can we identify itemsets that are emerging (submerging)? 2) Which of these itemsets have the potential to emerge (submerge) within the next time window? That is, we focus on finding emerging frequent itemsets in this paper.

According to the emerging itemsets of incremental scheme, we develop this concept on the temporal data mining. Temporal data mining has the limitation on window size for finding emerging itemsets. Therefore, we will change the formula for finding emerging itemsets in the following discussions.

Definition 2.1 *db_k is the transactions in t=k, e.g., db_1 is the transactions in t=1.*

Definition 2.2 *$DB_{i,i+1,\ldots,j}$ is the transactions in t=i to j, e.g., DB_{12345} is the transactions in t=1 to 5. We also view DB_{12345} as the accumulation of $db_1+db_2+db_3+db_4+db_5$.*

As an example, suppose original database is DB_{1234} and we set the limitation of window size as 5 as shown in Figure 2. Due to the limitation of window size, when adding a database db_6, we should discard the old database db_1. Hence, we must find potential emerging frequent itemsets from the database DB_{2345} before adding a new database db6 to form DB_{23456}, and this conforms the limitation of window size.

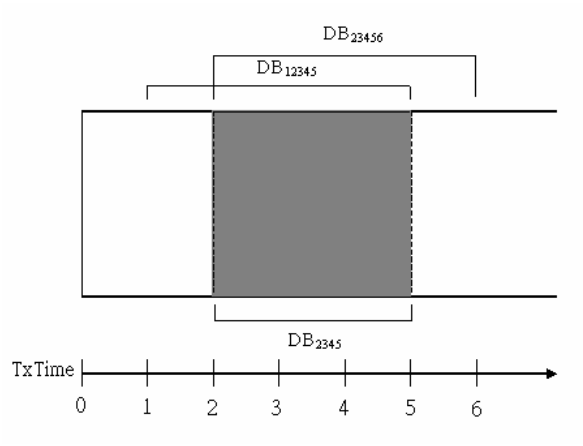

Figure 2. Potentially emerging frequent itemsets in DB_{2345}.

3: Mining Temporal Emerging Itemsets

In Section 3.1, we give an example for mining temporal emerging itemsets from data stream. The proposed algorithm, *EFI-Mine*, is described in details in Section 3.2.

3.1: Example for mining emerging itemsets

Figure 3 shows an example of emerging itemsets modified on that proposed by Dong and Li in [4] for the special case of EFI. It shows partitions of the space of itemsets from original database DB to the new database DB+db. Figure 3 plots the support count in DB (denoted as SC_{DB}) against the support count in db (denoted as SC_{db}). Each point in the graph depicts an ordered pair (SC_{db}, SC_{DB}) where the sum of SC_{db} and SC_{DB} is an itemset's support count in DB+db at some increment interval. If the increment adds no transactions to an itemset's support count, then its support count in DB has to be equal to $minSC_{DB}+minSC_{db}$ in order to achieve $minSC_{DB+db}$. This corresponds to point H in Figure 3. Alternatively, if an itemset's SC is equal to |db| in db,

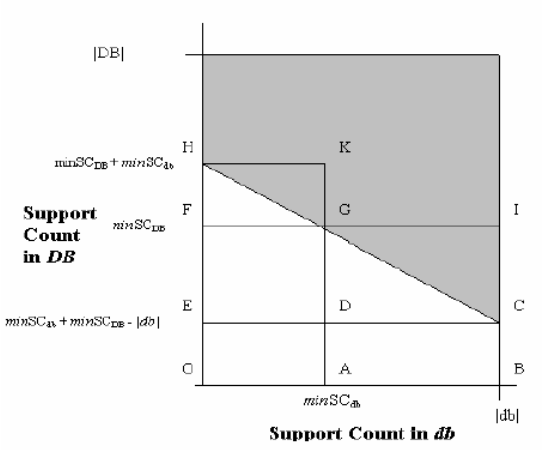

Figure 3. Emerging frequent itemsets.

then its support in DB has to be some SC=n, where n>0, and n= $minSC_{DB}$+$minSC_{db}$ -|db| for the itemset to be frequent.

This is point C in Figure 3. Line HC partitions the space of all itemsets in DB+db into frequent and infrequent. The shaded area in Figure 3 represents all the frequent itemsets and it includes Line HC. Specific partitions under HC contain itemsets that are emerging in the current increment. For example, the area defined by ΔHFG represents those itemsets that were frequent itemsets in DB, infrequent itemsets in db, and now are infrequent in DB+db. These itemsets have therefore submerged. ΔGIC represents itemsets that were infrequent in DB and frequent in db. These itemsets have emerged. Therefore, we can find all itemsets in area ABCG are emerging in the current interval and all itemsets in area OAGH are submerging.

However, there are too many emerging itemsets in area ABCG. In fact, to emerge in the next increment, the support count of an itemset in DB+db needs to be greater than or equal to 2$minSC_{db}$+$minSC_{DB}$ - |db| in the current increment. All points with this value are represented by line RS in Figure 4. The band of itemsets between line RS and line HC are all itemsets that have the potential to become frequent in the next increment, by this formula. Intersecting area ABCG and HCSR, we get itemsets in GDSC are most likely to emerge in the next increment.

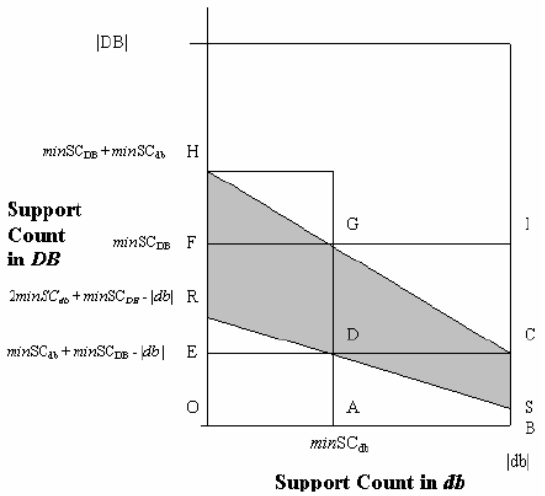

Figure 4. Potentially emerging frequent itemsets.

3.2: Algorithm of EFI-Mine

With window size we mention in Section 2.2 and the concepts of emerging itemsets in section 3.1, we set support value as S and assume the original database as $DB_{i,i+1,\ldots,j-1}$. According to the scheme we mentioned previously, if we want to find frequent itemsets from $DB_{i+1,i+2,\ldots,j+1}$, we should focus on $DB_{i+1,i+2,\ldots,j}$ for finding potential emerging frequent itemsets after adding database db_j and then find potential emerging frequent itemsets of the database $DB_{i+1,i+2,\ldots,j+1}$ before adding next incremental new database db_{j+1}. It means db_i would be an old database that needs not be considered. After adding new database db_{j+1}, the new database would be $DB_{i+1,i+2,\ldots,j+1}$. So the window size is N when database is changed from db_{i+1} to db_{j+1}. It also indicates N=(j+1)-(i+1)+1. By the feature of temporal data mining, we set $|db|=|db_i|=|db_{i+1}|=\ldots=|db_j|$. In Figure 4, various lines bear the following meaning:

$$LineHC = \min SC_{DB_{i+1,i+2,\ldots,j-1}} + \min SC_{db_j}$$

$$LineFI = \min SC_{DB_{i+1,i+2,\ldots,j-1}}$$

$$LineRS = 2\min SC_{db_j} + \min SC_{DB_{i+1,i+2,\ldots,j-1}} - |db_j|$$

$$LineEC = \min SC_{db_j} + \min SC_{DB_{i+1,i+2,\ldots,j-1}} - |db_j|$$

$$LineAK = \min SC_{dbj}$$

According to the feature of window size in temporal mining, incremental database means adding length of original transactions and also promoting the probability of infrequent itemsets to become frequent. Because we focus on N-1 window size for finding potential emerging frequent itemsets, these formulas should be divided by N-1 base on the number of database as follows:

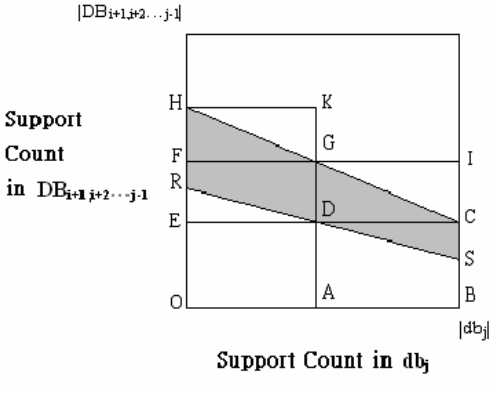

Figure 5. Potentially emerging frequent itemsets for temporal patterns.

$$LineHC = (\min SC_{DB_{i+1,i+2,\ldots,j-1}} + \min SC_{db_j}) / N - 1$$

$$LineRS = (2\min SC_{db_j} + \min SC_{DB_{i+1,i+2,\ldots,j-1}} - |db_j|)/\text{N-1}$$

Because line FI does not add new database, it should be divided by (N-1)-1. It means line FI should be divided by N-2 as follows:

$$LineFI = \min SC_{DB_{i+1,i+2,\ldots,j-1}} / \text{N-2}$$

Line EC means that adding new database db_j and an itemset's SC is equal to $|db_j|$ in db_j, so it should be divided by (N-1) as follows:

$$LineEC = (\min SC_{db_j} + \min SC_{DB_{i+1,i+2,\ldots,j-1}} - |db_j|)/\text{N-1}$$

Because db_j belongs to one of N window size, the formula should be divided by N as follows:

$$LineAK = \min SC_{dbj} / N$$

Figure 5 illustrates the potentially emerging frequent itemsets in area GDSC with window size limitation. The formula for each line is as mentioned above.

According to these formulas, we can simplify these lines as follows:

HC=[S*(j-1-(i+1)+1)*|db|+S*|db|]/N-1=
[S*(N-2)*|db|+S*|db|]/N-1= S*|db|

FI= [S*(j-1-(i+1)+1)|db|]/N-2= S*|db|

RS=[2*S*|db|+S*[(j-1)-(i+1)+1]*|db|-|db|]/N-1=
[2*S*|db|+S*(N-2)*|db|-|db|]/N-1= [(S*N)-1]*|db|/N-1

EC=[S*|db|+S*[(j-1)-(i+1)+1]*|db|-|db|]/N-1=
[S*|db|+S*(N-2)*|db|-|db|]/N-1= [S*(N-1)-1]*|db|/N-1

AK=S*db/N

Figure 6 shows the algorithm of *EFI-Mine* and the processing procedure is outlined below. The basic processing procedure is like Apriori except the definition of for minimum support value for finding temporal emerging itemsets from data stream. With window size N, we would not only remove db_i but also add new database db_j for finding 1-emerging itemsets on the database $DB_{i+1,i+2,\ldots,j}$ and finding large 1-itemsets on the database db_j from Step 1 to Step 3. So the purpose is to find potential emerging frequent itemsets of the database $DB_{i+1,i+2,\ldots,j+1}$ before adding next new database db_{j+1}. We generate k-candidates and find k-emerging itemsets by calculating support count as mentioned previously from Step 4 to Step 13. Then, we generate k-candidates and find k-large itemsets by support count we mention from Step 14 to Step 23. Finally, those itemsets meeting the constraints S*|db|> c.count ≧ [(S*N)-1]*|db|/N-1 on $DB_{i+1,i+2,\ldots,j}$ and c.count ≧S*db/N db_j are obtained as the potentially emerging frequent itemsets.

1) Input: $DB_{i,i+1,\ldots,j-1}$
2) Remove db_i from $DB_{i,i+1,\ldots,j-1}$ and add new database db_j then the database becomes $DB_{i+1,i+2,\ldots,j}$
3) E_1 = {emerging 1-itemsets on database $DB_{i+1,i+2,\ldots,j}$} and L_1={large 1-itemsets on database db_j};
4) **for** (k = 2; $E_{k-1} \neq 0$ k++) do begin
5) C_k = candidate-gen(C_{k-1}); // New candidates
6) **forall** transactions t ∈ $DB_{i+1,\ldots,j}$ do begin
7) C_t = subset(C_t, t); // Candidates contained in transactions
8) **forall** candidates c ∈ C_t do
9) c.count++;
10) **end**
11) E_k = {c∈ C_k | S*|db|> c.count ≧ [(S*N)-1]*|db|/N-1 }
12) // SC between LineHC and LineRS
13) **end**
14) **for** (k = 2; $L_{k-1} \neq 0$ k++) do begin
15) C_k = candidate-gen(C_{k-1}); // New candidates
16) **forall** transactions t ∈ db_j do begin
17) C_t = subset(C_t, t); // Candidates contained in transactions
18) **forall** candidates c ∈ C_t do
19) c.count++;
20) **end**
21) L_k = {c∈ C_k | c.count ≧ S*db/N }
22) // SC larger than LineAK
23) **end**
24) Output: $(\cup_k E_k) \cap L_k$;

Figure 6. Algorithm of *EFI-Mine*.

4: Experimental Evaluation

To evaluate the performance of *EFI-Mine*, we conducted experiments of using synthetic dataset generated via a randomized transaction generation algorithm in [6]. The synthetic data generation program takes the parameters as shown in Table 2, and the values of parameters used to generate the datasets are shown in Table 3. The main performance metrics used are execution time and accuracy. The accuracy is to measure the number of actual emerging frequent itemset in ratio of the total potential emerging frequent itemsets and defined as follows:

Accuracy = (number of actual emerging frequent itemset) / (total potential emerging frequent itemsets)

4.1: Effects of Varying Support Threshold

Table 2 Parameters of the synthetic datasets.

N	Number of items
T	Average numbers of items per transaction
C	Number of customers
D	Number of transactions
W	Windows size
S	Support value

Table 3 Parameter settings of synthetic datasets.

Dataset Parameters	N	T	C	D	W
N100T5C1000	100	5	1000	100,000	10

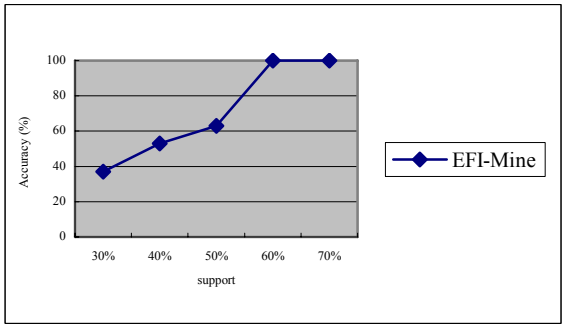

Figure 7. Accuracy under different support values (N100T5C1000, W=10).

In this experiment, we vary the values of support threshold from 30% to 70% for interesting the effects on the accuracy. The other parameters were kept fixed as default values. Figure 7 shows the accuracy of *EFI-Mine* under different support threshold values. It is observed that the average accuracy of potential emerging frequent itemsets raises as the support value is increased. Especially, the accuracy reaches to 100% when the support value is beyond 60%. Hence, *EFI-Mine* is verified to be very effective in finding the emerging itemsets.

4.2: Comparisons with Apriori

In this experiment, we compare the average execution time in different support values between Apriori and *EFI-Mine*. Both of these two algorithms could find frequent itemsets. However, Apriori can only find frequent itemsets, while *EFI-Mine* can find frequent itemsets that were infrequent in the past. Apriori algorithm processes $DB_{i+1,i+2,\cdots,j+1}$ to find frequent itemsets, while our *EFI-Mine* algorithm needs to process fewer database $DB_{i+1,i+2,\cdots,j}$ to find potentially emerging frequent itemsets. From Figure 8, *EFI-Mine* spends few seconds with high stability for finding potentially emerging frequent itemsets. Compared to Apriori, the improvement is about 90.6% for support values varied from 30% to 60%. Although *EFI-Mine* does not always obtain frequent itemsets with 100% accuracy, it reduces substantially the time in finding frequent itemsets. Moreover, the frequent itemsets obtained by Apriori are not suitable for applications in data streams since we need frequent itemsets which are infrequent in the past

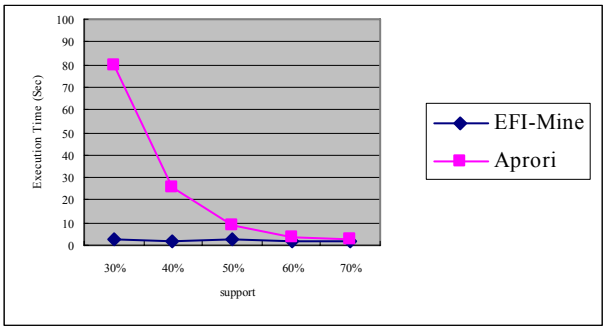

Figure 8. Execution time with w=10.

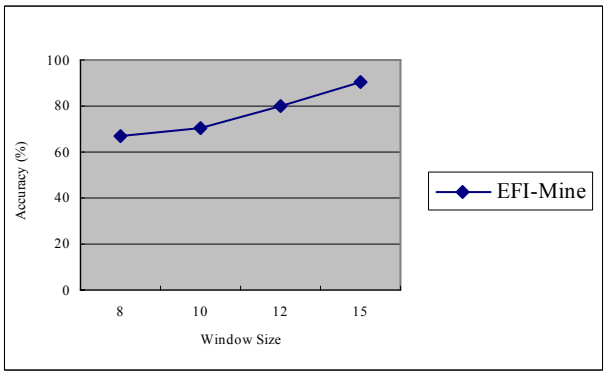

Figure 9. Accuracy under different window sizes.

and frequent in the current by the time change. Hence, *EFI-Mine* meets the requirements of high efficiency and high scalability in terms of execution time for data stream mining.

4.3: Effects of Varying Window Size

In this experiment, we investigate the effects of varying window size on the accuracy of mining results. As shown in Figure 9, we could observe that the larger window size, the higher with accuracy. In fact, the accuracy is almost 100% when window size is large than 15 in the experiments. This is because the itemsets are tended to be stable according to the past databases. This indicates that *EFI-Mine* fits for mining data streams with large window size.

4.4: Effects of Varying Transaction Size

In this experiment, we investigate the effects of varying transaction size on the accuracy of mining results i.e., the average number of items per transaction. As shown in Figure 10, if T is larger, the accuracy is higher than under T. This is because T can bring more information and trend from past transactions. This indicates that *EFI-Mine* fits for mining data streams with large transaction size.

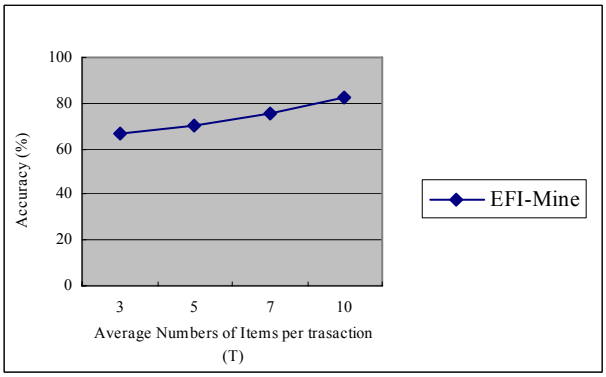

Figure 10. Accuracy under different numbers of items per transaction with w=10.

5: Related Work

In association rules mining, Apriori [1], DHP [9], and partition-based [10] are proposed to find frequent itemsets. Some algorithms like FUP [11], FUP_2 [12] and UWEP [13] are further proposed for finding frequent

itemsets in incremental databases. In recent years, a number of algorithms like FTP-DS [3] and RAM-DS [14] are proposed to process data in data streams. FTP-DS is a regression-based algorithm to mine frequent temporal patterns for data streams, while RAM-DS is a wavelet-based algorithm.

Dong and Li define an emerging pattern as an itemset the support of which increases significantly between two databases. We view emerging frequent itemsets as a special case of the emerging patterns described by Dong and Li. Recently, a new algorithm modifies an existing incremental algorithm, UWEP, so that it can identify emergent large itemsets. It uses incremental scheme for finding emerging frequent itemsets [5].

6: Conclusions

In this paper, we propose a new approach, namely *EFI-Mine*, which can discover emerging frequent itemsets from data streams efficiently and effectively. The novel contribution of *EFI-Mine* is that it can effectively identify the potential emerging itemsets such that the execution time can be reduced substantially in mining all frequent itemsets in data streams.

The experimental results show that *EFI-Mine* can find the emerging frequent itemsets with high precision under different conditions like varied window size, transaction size and number of items, etc. Hence, *EFI-Mine* promising for mining temporal emerging patterns in data streams.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In Proceedings of 1993 ACM SIGMOD Intl. Conf. on Management of Data, pages 207--216, Washington, D. C., May 1993.

[2] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo.Fast discovery of association rules. In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramaswamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 307--328. AAAI/MIT Press, 1996.

[3] W.-G. Teng, M.-S. Chen, and P. S. Yu. A Regression-Based Temporal Pattern Mining Scheme for Data Streams. Proceedings of the 29th International Conference on Very Large Data Bases, pages 93—104, September 2003.

[4] Guzhu Dong, Jinyan Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999.

[5] Susan P. Imberman, Abdullah Uz Tansel. and Eric Pacuit. An Efficient Method For Finding Emerging Frequent itemsets. 3rd International Workshop on Mining Temporal and Sequential Data, pages 112—121 August 2004.

[6] R. Agrawal and R. Srikant. Mining Sequential Patterns. Proceedings of the 11th International Conference on Data Engineering, pages 3—14, March 1995.

[7] David Wai-Lok Cheung, Sau Dan Lee, and Benjamin Kao. A general incremental technique for maintaining discovered association rules. In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997.

[8] Lin, C.H., Chiu, D.Y., Wu, Y.H., Chen, A.L.P. Mining Frequent Itemsets from Data Streams with a Time-sensitive Sliding Window. In Proceedings of 2005 SIAM International Conference on Data Mining, CA, April 2005.

[9] H. Cheng, X. Yan, and J. Han, IncSpan: Incremental Mining of Sequential Patterns in Large Database. Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining (KDD'04), Seattle, WA, Aug. 2004.

[10] S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In Proc. of the 8th Int. Conf. on Information and Knowledge Management (CIKM'99), Nov 1999.

[11] D. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. Proc. of 1996 Int'l Conf. on Data Engineering, pages 106—114, February 1996.

[12] D. Cheung, S.D. Lee, and B. Kao. A General Incremental Technique for Updating Discovered Association Rules. Proc. International Conference On Database Systems For Advanced Applications, April 1997.

[13] N.F. Ayn, A.U. Tansel, and E. Arun. An efficient algorithm to update large itemsets with early pruning. Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999.

[14] Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu: Resource-Aware Mining with Variable Granularities in Data Streams. SDM 2004.