Mining Closed-Regular Patterns in Incremental Transactional Databases using Vertical Data Format

M. Sreedevi
K L University
Vaddeswaram, Guntur
Andhra Pradesh, India

L. S. S. Reddy, Ph.D
LBR College of Engineering
Mylavaram, Vijayawada
Andhra Pradesh, India

ABSTRACT
Regular pattern mining on Incremental Databases is a novel approach in Data Mining Research. Recently closed item set mining has gained lot of consideration in mining process. In this paper we propose a new mining method called CRPMID (Closed-regular Pattern Mining on Incremental Databases) with sliding window technique using Vertical Data format. This method generates complete set of closed-regular patterns with support and regularity threshold values. Our Experimental results show that CRPMID method is efficient in both memory usage and execution time.

General Terms
Algorithms

Keywords
Closed patterns, Regular patterns, Vertical Data, Sliding Window, Incremental Databases.

1. INTRODUCTION
The importance of data mining is increasing rapidly. In many domains the contents of data is increasing with varying rates. In order to get interesting patterns old data is not adequate and updated data also to be considered. Recent many real life applications have called for the need of incremental mining. Mining incremental databases is one of the most needful areas in data mining research. This is due to the increasing usage of record based databases whose data increases continuously. Web content data, web usage data, stock market data, transactions in banking and retail marketing are some of the examples for incremental databases. A pattern is *regular* when it occurs at regular intervals in a database at a user given regularity threshold. A *regular* itemset is *closed-regular* if none of its immediate supersets has the same support as the *regular* itemset. Closed item set mining has been one of the searing areas in data mining research during the last decade. The literature [7, 8] have shown that closed item set mining is more desirable than mining complete set of frequent item set mining. For example in super market transactions It is required to know not only how frequently the items are moving and how regularly the items are moving also.

Regular pattern mining [1, 2] is also one of the recent thrust areas in data mining. A few algorithms are proposed so far to mine regular patterns in incremental databases. Many algorithms have been proposed so far to mine closed frequent patterns in incremental databases. To the best of our knowledge there is no suitable algorithm to mine closed-regular patterns in incremental databases. So, in this paper we propose a new algorithm called CRPMID with sliding window technique using vertical data format. In the process, first it mines for length-1regular itemset and then it finds length-2 regular itemset from the length-1 regular itemset and then the algorithm mines for length-1 closed-regular itemset. If an itemset is closed, automatically the itemset is closed frequent. Therefore mining closed-regular means mining closed frequent regular itemset.

The remaining of this paper is organized as follows. Section 2 describes related work, Section 3 describes problem definition, section 4 describes the process of closed-regular pattern mining, section 5 describes experimental results and finally section 6 will conclude the paper.

2. RELATED WORK
Researchers have recently explored the concept of mining frequent closed item sets than mining frequent item sets for discovering new surplus association rules [14, 15]. Mining frequent closed item sets instead of frequent item sets saves computation efforts and memory usage. Pasquier N et al., proposed [10] that the set of frequent closed item sets have been shown complete loss-less and reduced representation of frequent item sets.

Liu et al., have been proposed an algorithm for discovering frequent closed item sets in land mark window model. This algorithm works in batch mode, dividing the landmark window into several basic windows. In this process it ignores the recency of discovered item sets. Several algorithms like CLOSET, CHARM, Closet+, FP-Close, DCI-Closed, CHARM-L [13] have been proposed frequent closed item sets in static data sets. Takeaki et al., [11] proposed an efficient method to mine closed frequent item sets by constructing a tree that consists of only closed frequent item sets, but it consider all transactions mean while for finding these closed frequent item sets. Chi et al., derived a new algorithm MOMENT: which mines closed frequent item set over stream sliding window. This algorithm maintains selected set of items dynamically which includes four types of nodes. The main drawback of MOMENT is it judges closed item set indirectly through node checking along with type of nodes. So it consumes more time for type of node checking. Similarly MOMENT store much more information also other than current closed frequent item sets, which consumes more memory.

Tanbeer et al., introduced Regular pattern tree (RP-Tree) [1] to mine regular patterns from transactional database. The authors constructed a highly compact tree structure called RP-tree with a support descending order and a pattern growth approach to mine regular patterns in a static database and they extended it to incremental databases by constructing IncRT-Incremental regular pattern tree [2] and an item header table called IncRT-table to find regular patterns in incremental databases. IncRT-tree is also a highly compact tree structure with support descending order adjusting nodes every time whenever the database updates and the item header table plays an important role in mining regular patterns using IncRT-tree

in incremental databases. IncRT table consisting of five fields (*i, r, t_l, m, p*): item name (*i*), the regularity of *i* (*r*), last *tid* where item *i* occurred (t_l), a modification indicator of *i* (*m*), and a pointer to the IncRT for *i* (*p*). Later than inserting all transactions into the IncRT, *r* for all items is calculated in the table by traversing the tree once. Each time the database is updated, modification indicator *m* will modify the one bit field and t_l changes to the recent tid where item *i* occurred. The node traversal pointers only visit each tail-node of the item and accumulate tids available in its tid-list in respective temporary arrays for every item from the tail node up to the root node. After traversal to the top-most item in the table, the complete list of *tids* for *i* are obtained in their respective temporary arrays. Then the *periods* of *i* are calculated to obtain regular patterns. Association rule mining on incremental databases with sliding window filtering is explained in [4]. Nan Jiang and Le Grunewald proposed [6] CFI- Compute closed item sets online directly and incrementally. Closed item sets are maintained using lexicographical ordered DIU tree. When new transactions arrive closure property is checked and associated closed item sets and their support information is also updated. CFIM-Mining closed frequent item sets by eliminating null transactions is derived in [9].

3. PROBLEM DEFINITION

Let I = $\{i_1, i_2, \ldots, i_m\}$ be a set of items. A set $X = \{i_1, i_2, \ldots, x_n\} \subseteq$ I, where $1 \leq i \leq n$ is called a pattern or an itemset and transaction database DB has T = (tid, *X*) is a tuple where *tid* is a unique transaction identifier and *X* is a pattern. The item set with k number of items is called k-sized item set over the database DB. Transaction window W = $t_1, t_2, \ldots, t_n$ is a set of continuous transactions, where t_1 is the first transaction and t_n is the last transaction. A transaction sliding window W is a transaction window which contains fixed number of transactions, where |W| is size of the Transaction sliding window. The transaction sliding window will slide forward for every transaction. Let W1 be a first window and W2 be the second window and so on. The support of item *X* in transaction sliding window is represented as sup-count(*X*) that is number of transactions containing the *X* as sub item.

3.1 Definition 1 (period of X)

Let t_j^x and t_{j+1}^x are two consecutive transaction-ids in window The number of transactions between t_j^x and t_{j+1}^x is defined as a period of X, say p^x where $p^x = t_{j+1}^x - t_j^x$, $j \in (1, |W|)$. We consider the first transaction is t_f which is null transaction i.e $t_f = 0$ and last transaction is t_l which is last transaction in the window. Period of item X is defined as the no of times item X appears in different transactions.

3.2 Definition 2 (Regular itemset)

An item set *X* is a regular item set if regularity of *X* is less than or equal to user given regularity threshold value i e., λ.

3.3 Definition 3 (Closed Regular itemset)

Let $X = \{x_1, x_2, \ldots, X_n\}$ be a set of regular itemset and $Y = \{y_1, y_2, \ldots, y_n\}$ be other set of regular items, where $X \subseteq Y$ i e., *X* is a sub set of *Y* and *Y* is a super set of *X*, support count of *Y* must not be greater than support count of *X* then *X* is called closed-regular item set.

The problem is to mine closed-regular patterns in incremental transactional databases with sliding window technique. The transactional Database DB, db+ denotes the set of added transactions to DB. The updated database denoted as UDB (DB∪db+). W is the transaction sliding window, |W| is size of the window, λ is maximum regularity threshold value, and ∂ is minimum support count of an item set.

4. MINING CLOSED-REGULAR PATTERNS

Let us consider the below table is our example incremental transactional database containing nine transactions in original database DB and two transactions in the increment database db+. In this example the sliding window size is nine. In the mining process our CRPMID algorithm contains two phases.

Table1. Transaction Database UDB

Tid	Itemset
1	a, b, c, d
2	a, b, f
3	a, c, d, e
4	b, d, e
5	a, c, d, e, f
6	b, c, d
7	a, d, e
8	a, b, c, d, e, f
9	a, b, c, e
10	b, c, e
11	a, d, e, f

W2 W1 DB db+

In the first phase our algorithm mines for regular patterns and in the second phase it mines for closed regular itemsets for each window. W1 window contains nine transactions i e., from first transaction to ninth transaction. In W2 it contains nine transactions i e., from second transaction to tenth transaction and in W3 window from third transaction to eleventh transaction and continuous in this manner as the database updates.

First, convert W1 into vertical data format i.e., (X, Tid) X is an item set and Tid is transactional id. Table 2 contains Item set and its corresponding transaction Id number which are arranged in vertical database format. Regular items are mined based on periodicity or appearance of an itemset in different transactions. Consider the maximum difference of an itemset in the transactions as the periodicity of that itemset. The maximum transaction difference of an itemset should be less than or equal to the user given regularity threshold i e., *max_reg* < = λ and then the itemset is considered as regular itemset. Table 2 shows the itemsets and their corresponding transactions where each itemset occurs in the transactions. After converting the database into vertical data format, periodicity i e., P^X is calculated for each item set. Let us consider the λ = 4 and support-count ∂ = 5. For simplicity we assume the first transaction as $t_{first} = t_0$ which is a null transaction and last transaction is $t_{last} = t_n$. Regularity of an item set is obtained from P^X which is maximum periodicity of that item set. Regularity of item is calculated based on transaction difference. For example, item set <a> is appeared in transactions <1, 2, 3, 5, 7, 8, 9>, item set <b> is appeared in transactions <1, 2, 4, 6, 8, 9>. The transaction difference of item set < a > is <1, 1, 1, 2, 2, 1, 1>and the transaction difference of item set < b > is <1, 1, 2, 2, 2, 1>. Maximum transaction difference is considered as max_reg.

Table2. Vertical Data Format for one item set

Items	Tids
a	1, 2, 3, 5, 7, 8, 9
b	1, 2, 4, 6, 8, 9
c	1, 3, 5, 6, 8, 9
d	1, 3, 4, 5, 6, 7, 8
e	3, 4, 5, 7, 8, 9
f	2, 5, 8

The max_reg of itemset < *a* > is 2 which is shown in table 3. The support count of each item set will also calculate in this table. For example, Support count of itemset *a* is 7. Support count of an item is the number of times the item appeared in the transaction window.

Phase I.

Input: DB, λ

Output: Set of Regular Patterns

Procedure:

1. Convert Horizontal DB to Vertical DB
2. Let $X_i \subseteq$ I a k-item set
3. $P^X_i = 0$ for all X_i
4. For each X_i
5. Find the period of X_i
6. $P^X_i = P^X_{i+1} - P^X_i$
7. $reg(X_i) = max(P^X_i)$
8. repeat
9. if $reg(X_i) <= \lambda$
10. X_i is regular item set
11. Else
12. Delete X_i

In Phase I we find regular item sets. We convert the horizontal database of W1 window into vertical database as in Table 2. X_i is a pattern, P^X_i is periodicity of item set which is initialize to zero for all item sets. We find regularity for each item set which is the maximum periodicity of item set in that sequence. If the regularity item set is less than or equal to λ then the item set is regular item set otherwise the item set is non regular item set which will be deleted. From the obtained regular itemsets we find whether these itemsets are closed or not in second phase. After obtaining closed-regular itemsets in phase 2 we go for length-2 regular itemsets in phase 1 to obtain length-2 closed-regular itemsets in phase 2. This process will continue for three item sets, four item sets and so on until no closed-regular patterns found in W1. The same process will continue for W2, W3, W4, … to obtain latest closed-regular patterns. The closed-regular item sets are mined from regular item sets. We consider support counts for regular one item sets X_i, regular two item sets X_j and so on. The support count of regular itemset X_i is not less than or equal to the support count of X_j then X_i is closed- regular item set otherwise X_i is not closed regular item set. The item sets < a >, < b >, < c >, < d >, < e > are closed-regular item sets because the support counts of these itemsets are not less than or equal to the support counts of their supersets (Table 4) respectively where as < f > is regular but not closed-regular item set because itemset < f > support count is not satisfied with the support measure i e., $\partial = 5$.

Table 3. W1 - P^X, Regularity, Support

Items	P^X	Reg	Sup
a	1, 1, 1, 2, 2, 1, 1	2	7
b	1, 1, 2, 2, 2, 1	2	6
c	1, 2, 2, 1, 2, 1	2	6
d	1, 2, 1, 1, 1, 1, 1, 1	2	7
e	3, 1, 1, 2, 1, 1	3	6
~~f~~	~~2, 3, 3, 1~~	~~3~~	~~3~~

So delete the item set < f > from the list.

Phase II

Input: regular item sets, ∂

Output: Complete set of closed-regular patterns

1. Let $X_i \subseteq$ I is a regular k-item set
2. Let $X_j \subseteq$ I is a regular k+m item set
3. m= 1, 2, 3, ……, n
4. $X_i \subseteq X_j$ for all i <= j
5. Find $S(X_i)$, support-count of X_i
6. Find $S(X_j)$, support-count of X_j
7. If $S(X_i) > S(X_j)$
8. X_i is closed-regular item set
9. Else
10. Delete X_i

Table 4 contains all length-2 itemsets along with their transaction –ids which are formed from length-1 regular itemsets < a >, < b >, < c >, < d >, < e >.

Table 4. Vertical Data Format for two item set

Items	Transaction_id
a, b	1, 2, 8, 9
a, c	1, 3, 5, 8, 9
a, d	1, 3, 5, 7, 8
a, e	3, 5, 7, 8, 9
b, c	1, 6, 8, 9
b, d	1, 4, 6, 8
b, e	4, 8, 9
c, d	1, 3, 5, 6, 8
c, e	3, 5, 8, 9
d, e	3, 4, 5, 7, 8

Item sets < (a, c), (a, d), (a, e), (b, d), (b, e), (c, d) (c, e) (d, e)> are regular two item sets and < (a, b),(b, c) > are not regular two item sets which are not satisfied the specified regularity threshold value which are shown in table 5. With the length-2 regular itemset we find length-3 itemsets in order to find out

support and periodicity to check how many of these two itemsets are closed-regular itemsets. The process continues until no closed regular itemsets found in W1. Similarly the whole process will continues to W2, W3 and so on to find out all latest closed-regular patterns.

Table 5. W1 with periodicity, regularity and support for two item set

Items	P^X	Reg	Sup
~~a, b~~	~~1, 1, 6, 1~~	~~6~~	~~4~~
a, c	1, 2, 2, 3, 1	3	5
a, d	1, 2, 2, 2, 1	2	5
a, e	3, 2, 2, 1, 1	3	5
~~b, c~~	~~1, 5, 2, 1~~	~~5~~	~~4~~
b, d	1, 3, 2, 2	3	4
b, e	4, 4, 1	4	3
c, d	1, 2, 2, 1, 2, 1	2	5
c, e	3, 2, 3, 1	3	4
d, e	3, 1, 1, 2, 1, 1	3	5

5. EXPERIMENTAL RESULTS

5.1 Execution Evaluation

In this section we produce our results. Since there is no algorithm to mine closed-regular item sets in incremental transactional data bases we only examine our experimental results which include conversion of horizontal database into vertical format and closed regular patterns mining process. We did the experiment on synthetic dataset (T10I4D100k) and real dataset (Kosarak) with our CRPMID algorithm. These data sets are frequently used in frequent pattern mining experiments which are developed at IBM Almaden quest research group and are obtained from the website http://cvs.buu.ac.th/mining/Datasets/synthesis_data/ and UCI Machine Repository(University of California-Irvine,CA). We developed our algorithm in Java and our system configuration is 2.66 GHz CPU with 2GB main memory running on windows XP.

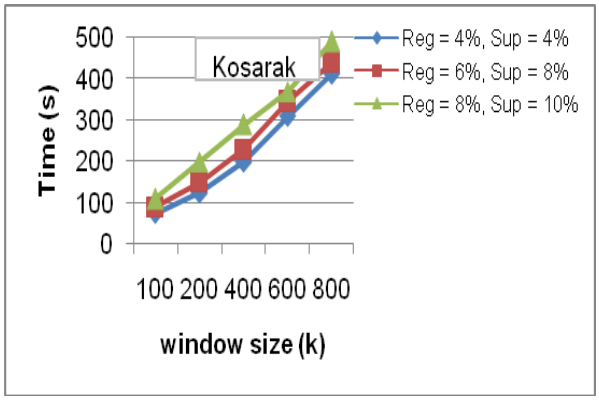

Figure 1: On Kosarak

In figure 1 we are showing the results on Kosarak dataset which contains 990K transactions, 41,270 items and 8.10 average transaction length. While regularity value 4% and support value 4%, the execution time is gradually increasing as the size of window increases. When regularity value is 6% and support value is 8%, the execution time is more when the window size is 400 and normal when window size is 800. This is because of vertical format and in same way when the regularity value is 8% and support value is 10% it takes more time at window sizes 400 and 600 and normal at the window size 800.

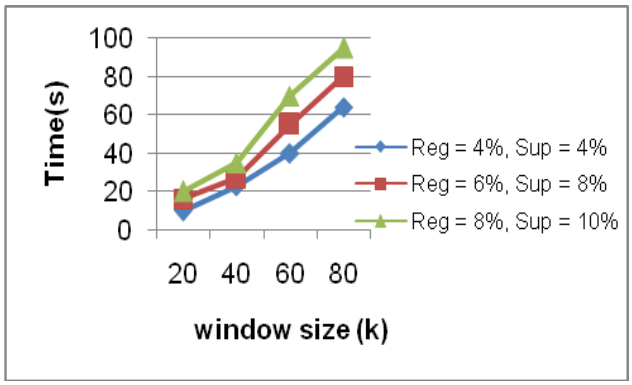

Figure 2: On T10I4D100K

We report the results on *T10I4D100K* synthetic dataset which contains 100K transactions, 870 items and 10.10 as the average transaction length. The above Figure 2 shows the execution time on different *Reg*() and *Sup*() values.

5.2 Performance Evaluation

The usage of memory by kosarak and T10I4D100K data sets is shown in this section. In Figure 3, we can see the memory usage when the regularity value is 60% and support value is 3% while executing on different database sizes with our algorithm.

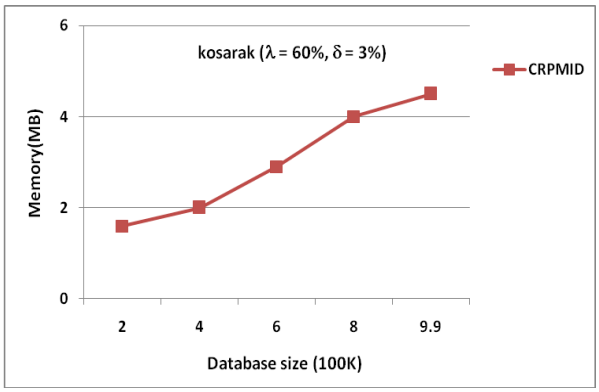

Figure 3: Kosorak

In Figure 4, we can see the memory usage when the regularity value is changing.

6. CONCLUSION

Closed regular pattern mining is a novel approach in data mining applications. We proposed CRPMID algorithm to mine closed regular patterns with regularity threshold and minimum support with sliding window technique in incremental transactional databases using vertical data format. The advantage of using vertical data format is it needs simple

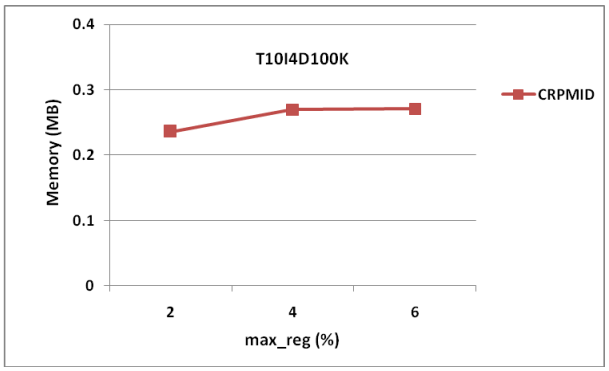

Figure 4: On T10I4D100K

Operations like union, intersection, deletion, arrays etc. Our experimental results show that the execution time over synthetic datasets and real datasets.

7. ACKNOWLEDGEMENTS

We are very thankful to Sri G.Vijay Kumar, Associate professor in Department of Computer science and Engineering , K L University, who supported and contributed towards development of our work.

8. REFERENCES

[1] Tanbeer and Ahmed. 2008. Regular pattern tree (RP-Tree) mines regular patterns from transactional databases. IECEC –Transactions Information and systems volume E91-D Issue-11, 2568-2577.

[2] Tanbeer and Ahmed. 2010 IncRT- Incremental regular pattern tree and pattern growth mining technique to find regular patterns on incremental databases. In Proceedings of International Asia pacific web conference.

[3] Vijay Kumar, G., Sreedevi, M., Pavan Kumar, N.,V.,S. 2011. Mining regular patterns on data streams using vertical data format. International Journal of Advanced Research in Computer Science. Volume 2, No. 5(Sept-Oct. 2011), 0976-5697.

[4] Chang-Hung Lee, Cheng–Ru Lin and Ming-Syan chen. 2001. Sliding–window Filtering: An efficient Algorithm for Incremental Mining. In Proceedings of the tenth international conference on Information and knowledge management. ACM.

[5] Chang and Lee. SWFI-Mining frequent item sets in online data streams with a transaction sliding windows model.

[6] Chi, Y., Wang, H., Yu, P. S., Muntz, R. R. 2004. Moment: Maintaining closed frequent itemsets over a stream sliding window. In Proceedings of Fourth IEEE International Conference on Data Mining.

[7] Zaki, M.,J., Hsiao, C.,J. 2002. CHARM: An Efficient Algorithm for Closed Item Set Mining. 2002. In Proceedings of SIAM International Conference on Data Mining.

[8] J.Pei, J.Han, and R.Mao. 2000. CLOSET Mining frequent closed item sets for Association Rules. In Proceedings of International Conference on Data Mining and Knowledge Discovery.

[9] Binesh Nair, Amiya Kumar and Tripathy. 2011. Accelerating Closed Frequent Item set Mining by Elimination of Null Transactions. Journal of Emerging Trends in Computing and Information Sciences. 2, 7 (July. 2011), 317-324.

[10] Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L. 1999. Efficient Mining of Association Rules using Closed Itemset Lattices, Journal of information Systems 24(1), 25-46.

[11] Uno, T., Asai, T., Uchida, Y., Hiroki A. 2003. LCM: Enumerating Frequent Closed Item sets. In Proceedings of the IEEE ICDM Workshop of Frequent Itemset Mining Implementations (FIMI).

[12] Liu,X., Guan J., Hu, P. 2009. Mining frequent closed item sets from a landmark window over online data stream. Journal of computers and Mathematics with Applications. 57, 6 (2009). 927-936.

[13] Han J., Cheng,H., Xin, D.,Yan. 2007. Frequent Pattern Mining: Current Status and future Directions. Journal of Data Mining and Knowledge Discovery.15 (2007) 55-86.

[14] Stumme, G., Taouil, R., Bastide, Y., Pasquier, N., Lakhal, L. 2002. Computing iceberg concept lattices with TITANIC. Journal on Data and Knowledge Engineering. 42, 2 (2002).189-222.

[15] Zaki, M.J. 2001. Generating Non-Redundant Association Rules. In Proceedings of ACM SIGKDD International Conference on knowledge Discovery and Data Mining.

[16] Yuan, D., Lee, K., Cheng, H., Krishna, G., Li, Z., Ma, X., Zhou, Y., Han, J. 2008. CISpan: comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining. In Proceedings of SIAM Int. Conf. Data Mining.

[17] Chen, Y., Guo, J., Wang, Y., Xiong, Y., & Zhu, Y. 2007. Incremental Mining of Sequential Patterns using Prefix Tree. Advances in Knowledge Discovery and Data Mining, 433-440.