Incremental Temporal Mining using Incremental TPMiner and Incremental P-TPMiner Algorithms

R. V. Argiddi
Assistant Professor
Department of Computer Science, WIT
Solapur University
Solapur-413006, India

Sonali Vijaykumar Rampure
P.G. Student
Department of Computer Science, WIT
Solapur University
Solapur-413006, India

ABSTRACT
Temporal data mining is one type of "predictive". Temporal Association mining is sequential mining. We usually predict what will happen next or what is probability that certain thing happen Sequential pattern mining is one important case of data mining. Most of sequential pattern mining algorithm works on static data which deals with the database should not change. But the databases in real world application do not have static data rather they have incremental database. There are some applications using temporal event data have used to discovering patterns from events. There are two types of interval-based patterns: Temporal pattern and Probabilistic temporal pattern are proposed. This paper attempts to provide two algorithms Incremental Temporal Pattern Miner (TP-Miner) and Probabilistic Temporal Pattern Miner (P-TP Miner).In this project, apply proposed algorithms to real datasets to make the comparison of Incremental temporal mining and Non-incremental temporal mining.

General Terms
Temporal Mining, Incremental Temporal Mining, Candidate generate.

Keywords
Sequential pattern, Incremental Temporal Pattern, Interval based pattern, Data mining.

1. INTRODUCTION
In many real time applications, sequence databases are updated with time incrementally. Based on that sequence database with time many sequential patterns are discovered. Re-mining sequential patterns from scratch each time is inefficient and not feasible if some new sequences are grow or added into the database. Incremental temporal mining which is nothing but the issue of maintaining that discovered patterns over time in the presence of more items being added into the database. Because of the mostly appending or updating time-series data, incremental mining would be very effective and efficient. Temporal Data Mining is the process of Knowledge Discovery in Temporal Databases that discovered temporal patterns over the temporal data, and Temporal Data Mining Algorithm is an algorithm that enumerates temporal patterns over temporal data .That temporal data is real time or synthesis data. By analyzing temporal data and finding temporal patterns is concerned with temporal data mining [10].Temporal databases are categorized by month, day of week or by season, time interval etc. Temporal databases are continuously updated or appended so that the discovered rules need to be updated. Re-running the temporal mining algorithm every time is inefficient since it ignores previously discovered patterns and repeats the work done previously. Incremental mining algorithms can significantly increase the speed of a task because much of the work that was performed for previous tasks can be reused in successive tasks. Since Incremental mining algorithms have more speed than non-incremental mining algorithm. Non-incremental algorithm may not be applicable for extremely large data sets but an incremental algorithm may be applicable. Therefore in this project proposed incremental Temporal mining (TP) and Probabilistic Temporal mining (P-TP).Incremental algorithms are faster than non - incremental because they re-mine their previous task and use the results .Both TP and P-TP are developed efficiently and they work effectively. Incremental mining algorithms can significantly increase the speed of a task because much of the work that was performed for previous calls tasks can be reused in successive searches. These algorithms are most advantageous when the successive tasks that the incremental algorithm is run on are similar to previous tasks. In this training data is presented one at a time. Incremental mining algorithms for mining frequent patterns that use information collected during earlier mining process to cut down the cost of finding new pattern in whole database. Since mining every time the database grows, it becomes inefficient and hence the algorithm for incremental mining has to be proposed.

2. LITERATURE REVIEW
Y. Li, J. Bailey, L. Kulik and J. Pei,[5] In this paper, there is pattern mining for uncertain sequences and introduces probabilistic frequent spatial-temporal sequential patterns with gap constraints. These patterns are important for the discovery of knowledge given data. They proposed a dynamic programming approach for computing the frequentness probability of these patterns, which has linear time complexity, and it is embedding into pattern algorithms using both breadth-first search (BFS) and depth-first search (DFS) strategies. Temporal sequential pattern concept and frequencies are helpful for this incremental temporal mining algorithm and probabilistic temporal mining algorithm. Linear complexity is helpful to many search algorithms. These strategies are related to data structure.

A. Wong, D. Zhuang, G. Li, and E. Lee,[7] This paper improves the output quality by removing two types of redundant patterns. The first concept is to remove redundant patterns that are not delta closed the notion of delta tolerance closed item set is employed. The second concept is to capture redundant patterns statistically induced patterns is proposed yet their significance is induced by their strong patterns.

Yi-Cheng Chen, Wen-Chih Peng and Suh-Yin Lee, [1] In this paper, two algorithms (TPMiner and P-TPMiner) are developed to discover temporal pattern, occurrence-probabilistic temporal pattern and duration-probabilistic temporal pattern using two novel representations endtime and

endpoint representation. These TP-Miner and P-TPMiner are used to determine patterns which are temporal based on real time dataset or synthesis dataset. Based on application choice of dataset is determined and algorithms applied on it.

Y. Chen, C. Chen, W. Peng and W. Lee, [2] In this paper, a novel algorithm, namely, Correlation Pattern Miner(CoPMiner), is developed to capture the usage patterns and correlations among appliances probabilistically effectively and efficiently. Search space is reduced by using CoPMiner. There are some optimization techniques which are based on CoPMiner used to minimize search space.

H. Kim, M. Marwah, M. Arlitt, G. Lyon and J. Han, [10] This paper results indicate that a conditional factorial hidden semi-Markov model which integrates additional features related to when and how appliances are used in the home and more accurately represents the power use of individual appliances, outperforms the other unsupervised disaggregation methods.

Y. Chen, J. Jiang, W. Peng and S. Lee, [12]In this paper, Mining temporal patterns from time interval based data is a difficult problem since processing of complex relations among intervals may require generating and examining large amount of intermediate subsequences. Therefore to remedy this issue incision strategy is a technique for coincidence representation.

J.Kolter, and M. Johnson, [3] In this paper, a public data set REDD that is The Reference Energy Disaggregation Data Set which is real time dataset used for many data mining applications. This REDD dataset is used to determine temporal pattern because dataset have power readings. Using this real time dataset algorithm can produce pattern based on start time and end time by applying temporal pattern algorithm.

3. METHODOLOGY

3.1 Association rules mining

It is intended to identify strong rules discovered in databases using two different measures of interestingness. The first one is support which generates frequent item set from the provided database and the other one is confidence which is focuses on rule generation. In this project support count for frequent item sets are calculated and used re-mining next item sets. Calculated support count is used to determining pattern support count

3.2 Frequent item sets

A set of attributes is termed as frequent item set if the occurrence of the set within the database is more than a user given threshold. Support and Confidence terms are described below using its formula.

Support- Support determines how often a given rule is applicable to a given data set.

$Support$, s (X->Y) = $(X\ U\ Y)\ N$

Confidence- Confidence determines how frequently items in Y appear in transactions that contain X.

$$Confidence,\ c(X->Y) = \frac{\sigma(X\ U\ Y)}{\sigma(X)}$$

The proposed system Fig I. shows that association rule mining process will apply on original data and generated data will stored in intermediate data. This association rules mining process will also produces original pattern. Incremental association rule mining process will apply on incremental data and generated data will stored in intermediate data. This Incremental association rules mining process will produces updated pattern based on available knowledge (obtained from mining of previously Intermediate stored data) and original pattern. The incremental mining algorithm uses incremental mining technique is to re-run the mining algorithm on the only updated database. The proposed algorithm Incremental Temporal Pattern Miner first transforms the temporal database into the endpoint representation for determining patterns.

It then read the intermediate results from InetrmediateDataManager. InetrmediateDataManager saves intermediate result to reuse it in incremental mining. Next, it will scan the database to calculate the support count of each endpoint concurrently. Users are taking threshold as input from user. Incremental TPMiner removes infrequent endpoints below the given minimum threshold which is entered from user. For each frequent starting endpoint ,build the projected database and call minePatterns recursively to discover sets of all temporal patterns. Save intermediate result to reuse it in incremental mining so that it writes intermediate results in InetrmediateDataManager.

In minePattern, for a prefix it scans its projected database once to discover all frequent endpoints and remove infrequent ones. It computes support for frequent endpoint .Frequent endpoint can be appended to the original prefix to generate a new frequent sequence. If all endpoints in a frequent endpoint sequence appear in pairs every starting (finishing) endpoint has a corresponding finishing (starting) endpoint, now output this frequent endpoint sequence. Use calculated support count to remine previous count support. Finally, construct the projected database with the frequently extended prefixes and recursively call minePattern until the prefixes can no longer be extended.

Algorithm: TPMiner

Input : DB is Temporal Database ,**min_threshold** is minimum support threshold

Output: TP : Temporal Pattern, Support

1. Get real time database for incremental temporal mining.
2. Read Intermediate results from IntermediateDataManager.
3. Transform the *DB* into endpoint representation.
4. Find all frequent endpoints in *DB.*
5. Call *minePatterns* (*DB*).
6. Call *ComputeSupport*(*DB*, patterns)
7. Calculate *frequency* of endpoints and remove infrequent endpoints in databse those are less than
8. *min_threshold*.
9. *FE* <- all frequent "starting endpoints";
10. *for* each *s Є FE* do
11. Generate candidates DB|@;
12. Call TPMiner algorithm
13. Produce *TP*
14. Write an Intermediate Result.
15. *end*

Procedure *minePatterns* (@, DB|@)

16. for each *s Є FE* do
17. append s to @ to form @'
18. if @' is a temporal pattern then // if all endpoints appear in pair in @'
19. *TP ← TP ∪ @' ;*
20. *DB|@' ← DB_construct (DB|@, @'); //*

21. *ComputeSupport* (*DB|@ , patterns*); // calculate support count
22. *end*

Procedure *ComputeSupport (DB|@ , patterns)*

for each endpoint sequence q ∈ DB|@ do
23. mark a "stop_position" at the first (leftmost) finishing endpoint which has corresponding starting endpoint in @ ;
24. Count the support of every endpoint from the beginning of q to the stop_position of q.
25. *end*

Procedure *Generate Candidates*

26. temporal_seq ← □;
27. find all postfix sequences of @' in DB|@ to form DB|@' ;
28. *for* each postfix sequence q ∈ DB|@' do
29. eliminate the "finishing points" in q which has no corresponding "starting point" in @' ; // postfix-pruning strategy
30. temporal_seq ← temporal_seq ∪ q;
31. return temporal_seq
32. *end*

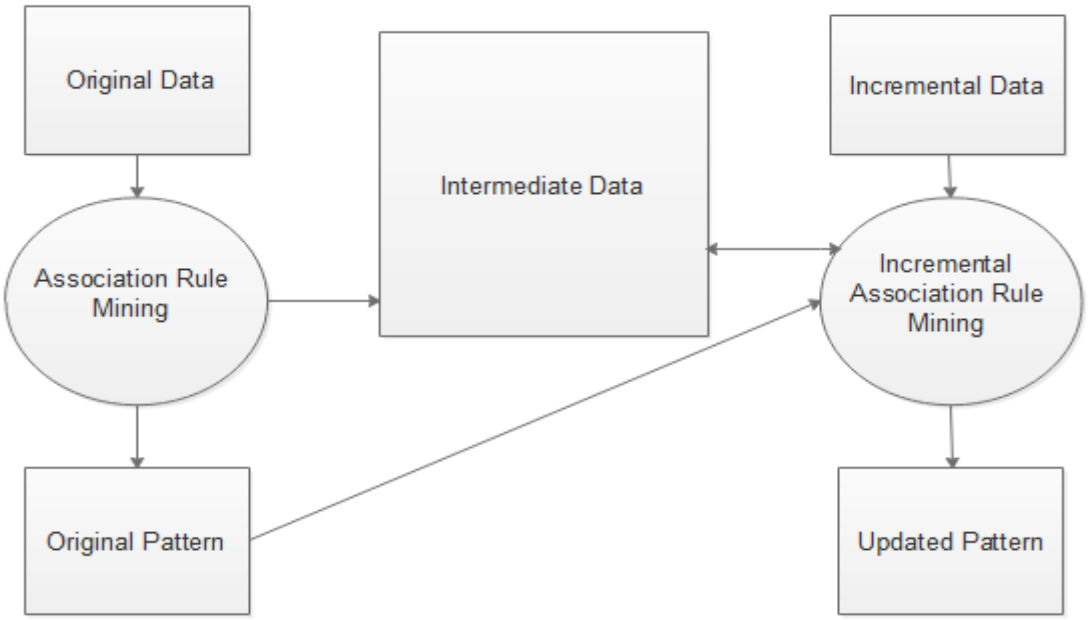

Fig 1 Flow of proposed work

The proposed algorithm Incremental Probabilistic Temporal Pattern Miner (P-TPMiner) first transforms the temporal database into the endpoint representation. It then read the intermediate results from InetrmediateDataManager. InetrmediateDataManager saves intermediate result to reuse it in incremental mining. Next, it will scan the database to calculate the support count of each endpoint concurrently. Users are taking threshold as input from user. Incremental TPMiner removes infrequent endpoints below the given minimum threshold which is entered from user. For each frequent starting endpoint now build the projected database and call minePatterns recursively to discover sets of all temporal patterns. In this project save intermediate result to reuse it in incremental mining so that it writes intermediate results in InetrmediateDataManager.

In minePattern, for a prefix it scans its projected database once to discover all frequent endpoints and remove infrequent ones.It compute support for frequent endpoint .Frequent endpoint can be appended to the original prefix to generate a new frequent sequence. If all endpoints in a frequent endpoint sequence appear in pairs every starting (finishing) endpoint has a corresponding finishing (starting) endpoint, now output this frequent endpoint sequence. Use calculated support count to remine previous count support. Finally, construct the projected database with the frequently extended prefixes and recursively call minePattern until the prefixes can no longer be extended.

The proposed algorithm Incremental Probability Temporal Pattern Miner (P-TPMiner) first transforms the temporal database into the endpoint representation. It then read the intermediate results from InetrmediateDataManager. Now, save intermediate result to reuse it in incremental mining. Next, it will scan the database to calculate the support count of each endpoint concurrently. Users are taking threshold as input from user. Incremental P-TPMiner removes infrequent endpoints below the given minimum threshold. For each frequent starting endpoint build the projected database and call minePatterns recursively to discover sets of all temporal patterns. Save intermediate result to reuse it in incremental mining so that it writes intermediate results in InetrmediateDataManager. In minePattern, for a prefix it scans its projected database once to discover all frequent endpoints and remove infrequent ones. It compute support for frequent endpoint .Frequent endpoint can be appended to the original prefix to generate a new frequent sequence. If all endpoints in a frequent endpoint sequence appear in pairs

every starting (finishing) endpoint has a corresponding finishing (starting) endpoint, now output this frequent endpoint sequence. Use calculated support count to remine previous count support. Finally, construct the projected database with the frequently extended prefixes and recursively call minePattern until the prefixes can no longer be extended.

Algorithm: P-TPMiner

Input: DB is Temporal Database, **min_threshold** is minimum support threshold.

Output: TP : Temporal pattern,Support.

1. Get real time database for temporal mining.
2. Read Intermediate results.
3. Transform *DB* into endpoint representation.
4. Find all frequent endpoints.
5. Call *minePatterns* (*DB*).
6. Call *ComputeProbablities*(*DB*,patterns)
7. Calculate *frequency* of endpoints and remove infrequent endpoints in databse those are less than *min_threshold*.
8. *FE* <- all frequent "starting endpoints";
9. *for* each *s* ∈ *FE* do
10. Generate Candidates *DB|s*;
11. Call P-TPMiner
12. Produce *TP*
13. Write an Intermediate Result.
14. *end*

Procedure *minePatterns (@, DB|@)*

15. *for* each *s* ∈ *FE* do
16. append s to @ to form @'
17. if @' is a temporal pattern then // if all endpoints appear in pair in @'
18. *TP* ← *TP* ∪ *@' ;*
19. *DB|@'* ← *DB_construct (DB|@, @');*
20. *ComputeProbablities* (*DB|@ , patterns*); //calculate support count.
21. *end*

Procedure *ComputeProbablities (DB|@ , patterns)*

22. *for* each endpoint sequence q ∈ DB|@ do
23. Mark a "stop_position" at the first (leftmost) finishing endpoint which has corresponding starting endpoint in @ ;
24. Count the frequency of every endpoint from the beginning of q to the stop_position of q.
25. Calculate probability of every point representation .
26. *end*..

Procedure *Generate Candidates*

27. temporal_seq ← □;
28. find all postfix sequences of @' in DB|@ to form DB|@' ;
29. *for* each postfix sequence q ∈ DB|@' do
30. eliminate the "finishing points" in q which has no corresponding "starting point" in @' ; // postfix-pruning strategy
31. temporal_seq ← temporal_seq ∪ q;
32. return temporal_seq
33. *end*

4. EXPERIMENTAL RESULTS

Time complexity analysis and experimental results show that the incremental algorithm is superior to non-incremental algorithms when dealing with large data sets. Both algorithms are proposed this incremental strategy by saving intermediate results (IR).This IntermediateDataManager runs these incremental algorithms in restartable mode. For saving intermediate results IntermediateDataManager is used to run this algorithm in restartable mode. By extracting intermediate results we are mining previous results with next successive tasks. The Fig II shows that execution time requires for incremental mining is less than execution time require for non-incremental mining. Incremental temporal mining is applied on many real world applications. Testing is performed on real time dataset with minimum support threshold. The system is developed using Java platform. The below result analysis is comparison of Non-incremental temporal mining and Incremental temporal mining.

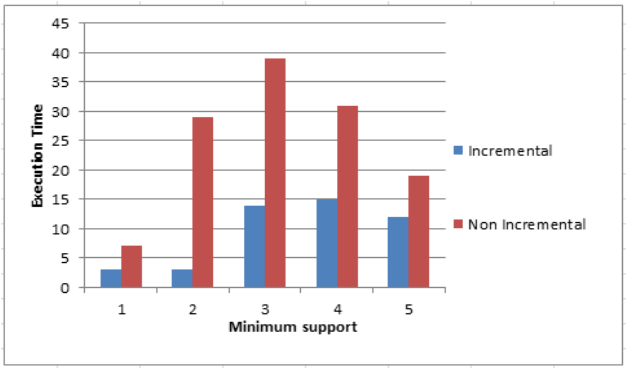

Fig.2 Result Analysis

5. CONCLUSION

We are discussed Incremental temporal mining with non-incremental algorithm. In this system we have proposed the incremental algorithms and how these algorithms are run in restartable mode. Both incremental Temporal Pattern (TP) and Probabilistic-Temporal Pattern (P-TP) are developed efficiently and effectively. These both algorithms are reusing previous results so that these algorithms don't require to start candidate generation for their previous tasks. Execution time requires for incremental mining is less than execution time require for non-incremental mining. The experimental results shows that overall performance of these two Temporal Pattern and Probabilistic-Temporal Pattern are better than existing TP and P-TP algorithms. Chronological order is an issue of incremental temporal mining.

6. ACKNOWLEDGMENTS

We are thankful to Dr. Mrs.S.S.Apte for providing necessary guidance concerning projects implementation.

7. REFERENCES

[1] Yi-Cheng Chen, Wen-Chih Peng and Suh-Yin Lee, "Mining Temporal Patterns in Time Interval Based Data" IEEE Transactions on Knowledge and Data Engineering, 1041-4347 (c) 2015 IEEE.

[2] Chen,C.Chen,W.Peng and W. Lee, "Mining Correlation Patterns among Appliances in Smart Home Environment," IEEE 18th Pacific-Asia Conference in Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining (PAKDD'14), pp. 210-221, 2014.

[3] J.Kolter, and M. Johnson, "REDD: A public data set for energy disaggregation research," KDD workshop on Data Mining Applications in Sustainability (SustKDD'11), pp. 1-6, 2011.

[4] S. van Schaik, D. Olteanu and R. Fink, "ENFrame: A Platform for Processing Probabilistic Data," The 17th International Conference on Extending Database Technology *(*EDBT'14*)*, pp. 355-366,2014.

[5] Y. Li, J. Bailey, L. Kulik and J. Pei, "Mining Probabilistic Frequent Spatio-Temporal Sequential Patterns with Gap Constraints from Uncertain Databases," The 13th International Conference on Data Mining *(*ICDM'13*)*, pp. 448-457, 2013.

[6] R.Sadasivam and K. Duraiswamy, "Efficient Method to Discover Interval-based Sequential Patterns," Journal of Computer Science, vol. 9, issue 2, pp. 225-234, 2013.

[7] A.Wong, D. Zhuang, G. Li, and E. Lee, "Discovery of Closed Patterns and Noninduced Patterns from Sequences," IEEE Transactions on Knowledge and Data Engineering, vol.24, no. 8, pp.1408-1421, 2012.

[8] A.Zakour, S. Maabout, M. Mosbah and M. Sistiaga, "Uncertainty Interval Temporal Sequences Extraction," International Conference on Information Systems Technology and Management (ICISTM' 12), pp. 259-270, 2012.

[9] Z. Zhao, D. Yan and W. Ng, "Mining Probabilistically Frequent Sequential Patterns in Large Uncertain Databases," The 15th International Conference on Extending Database Technology *(*EDBT'12*)*, pp. 74-85, 2012.

[10] H. Kim, M. Marwah, M. Arlitt, G. Lyon and J. Han, "Unsupervised disaggregation of low frequency power measurements," The 11th SIAM International Conference on Data Mining (SDM'11),pp. 747–758, 2011.

[11] M. Muzammal and R. Raman, "Mining Sequential Patterns from Probabilistic Databases," The 15th Pacific-Asia Conference in Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining, (PAKDD'11), pp. 210-221, 2011.

[12] Yi-Cheng Chen, Ji-Chiang Jiang, Wen-Chih Peng and Suh-Yin Lee,"An Efficient Algorithm for Mining Time Interval-based Patterns in Large Databases",(CIKM'10), October 26–30, 2010 2010 ACM.