Present State-of-The-ART of Dynamic Association Rule Mining Algorithms

N.Satyavathi, B.Rama,A.Nagaraju

Abstract*: Association Rule Mining (ARM) is a data mining approach for discovering rules that reveal latent associations among persisted entity sets. ARM has many significant applications in the real world such as finding interesting incidents, analyzing stock market data and discovering hidden relationships in healthcare data to mention few. Many algorithms that are efficient to mine association rules are found in the existing literature, apriori-based and Pattern-Growth. Comprehensive understanding of them helps data mining community and its stakeholders to make expert decisions. Dynamic update of association rules that have been discovered already is very challenging due to the fact that the changes are arbitrary and heterogeneous in the kind of operations. When new instances are added to existing dataset that has been subjected to ARM, only those instances are to be used in order to go for incremental mining of rules instead of considering the whole dataset again. Recently some algorithms were developed by researchers especially to achieve incremental ARM. They are broadly grouped into Apriori-based and Pattern-Growth. This paper provides review of Apriori-based and Pattern-Growth techniques that support incremental ARM.

Keywords: Data mining, incremental association rule mining, apriori-based, Pattern-Growth ARM algorithms.

I. INTRODUCTION

Enterprises in the real world are giving unprecedented importance to their business data. This shift in thinking on data has led to the faster growth of databases. Databases are subjected to frequent changes. When large databases are analyzed, it results in business intelligence or actionable knowledge which helps in making well informed decisions. Often association rule mining produces rules that can help in understanding the association or relationship among different entities or in the database. In fact, ARM is used for acquiring knowledge from databases in many areas such as banking, insurance, e-Commerce, e-Governance, and education and so on. In this context, ARM assumes significance and it needs wherewithal to have incremental capabilities to reduce complexity, time taken and resources consumed. In other words, ARM algorithms that are incremental are required to save time and effort besides make the new rules quickly thus minimizing convergence of business intelligence.

Incremental ARM has thus gained significance in rendering data mining service for more efficient means of obtaining Knowledge from large databases. The aim of it is to ensure the generation of rules based on the newly added records and update the existing rules in fairly less amount of time. Though it is incremental mining of rules, it needs to consider the knowhow already generated to have updated rules. Of late many algorithms came into existence to leverage ARM process in that way. They are broadly categorized into Apriori-based methods and Pattern-Growth ones. This paper provides the brief review of both apriori-based and pattern-Growth algorithms. The reminder of this paper is structured as follows. Section 2 provides the concept of incremental association rule mining. Section 3 and section 4 provides state-of-the-art on apriori-based incremental ARM and experimental analysis respectively, Section 5 and section 6 provides state-of-the-art on Pattern-Growth incremental ARM and experimental analysis respectively. Section 7 on the other hand provides conclusions besides giving directions for future work.

II. CONCEPT OF INCREMENTAL ASSOCIATION RULE MINING

In order to optimize the ARM algorithms in terms of reducing time and space complexity, the concept of incremental association rule mining came into existence. According to this concept, the mining takes place incrementally rather than reinventing the wheel every time. As new data is added to databases, the algorithm needs to update already mined rules just by using newly added records. The databases that exhibit constant arrival of new records are in the fields of weather forecasting, supermarket data, e-Commerce, mailing and so on. Therefore it is made inevitable to have incremental mining of association rules. The intend of such mining is to re-run the algorithm on the modified database instead of the whole database. The procedure by which the incremental ARM takes place is illustrated in Fig.1.

Revised Manuscript Received on November 05, 2019.

N.Satyavathi,CSE department, Vaagdevi college of Engineering, Warangal ,India. Email: satyanadendla15@gmail.com

Dr.B.Rama, Computer Science, Kakatiya University,Warangal, India. Email: rama.abbidi@gmail.com

Dr.A.Nagaraju, Computer Science,School of mathematics, statistics and computational science,Central University of Rajasthan, Ajmer, India. Email: nagaraju@curaj.ac.in

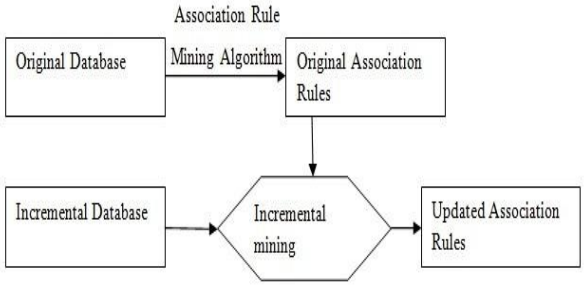

Fig.1. Process of incremental mining

The incremental ARM algorithms need to consider newly added transactions in the database and already mined rules for further processing. This will decrease computational complexity significantly. There are many incremental mining algorithms that came into existence, which are identified as two groups: Apriori-based and Pattern-Growth incremental mining algorithms.

III. APRIORI-BASED INCREMENTAL MINING ALGORITHMS

As mentioned earlier, there are many incremental ARM algorithms based on Apriori mechanism. The general procedure of Apriori algorithm is shown in Fig.2.

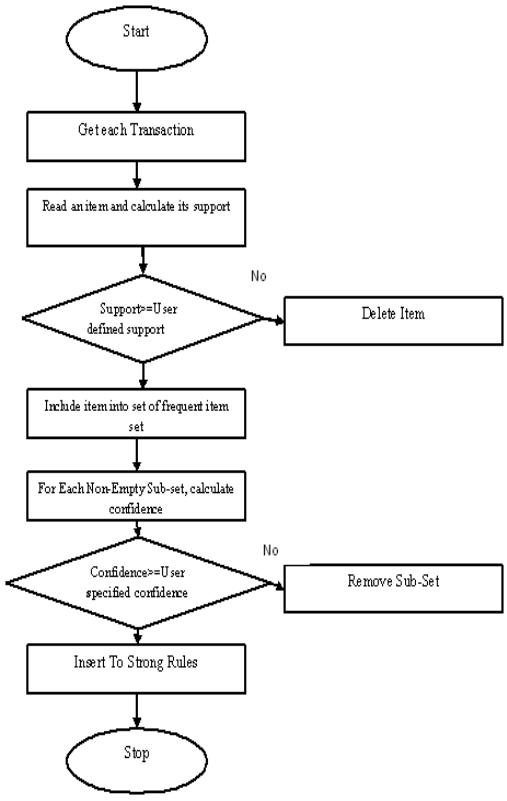

Fig.2.General procedure of Apriori Algorithm.

This section provides a brief overview of them.

A. FUP Algorithm

Fast UPdate (FUP) [1] is the algorithm that exhibits Incremental ARM. It supports incremental update of mined association rules in case of new records insertion. It can update the rules based on the changes made to database incrementally. The algorithm contains much iteration and each iteration generates a candidate set, subjected to the frequent itemsets already mined in the previous iteration.

B. FUP2 Algorithm

This algorithm is proposed by Cheung et al. [2], an extension to the existing FUP algorithm. As mentioned above FUP is able to generate association rules incrementally when latest transactions are added or existing ones are deleted. The FUP2, on the other hand, supports incremental ARM in case of both new record insertions and deletion of existing records.

C. Algorithm Utilizing Negative Borders

This algorithm [5] makes use of the notion of negative borders. Thus it can improve the performance of FUP-based algorithms. First of all, it generates frequent item sets related to an increment in the database D'. It goes for a full scan of the entire database D only when an item set is outside the negative border. In such cases only it needs one scan of the entire database. Its drawback is that the size of candidate dataset may be increased as it considers negative border closure.

D. *DELI (*Difference Estimation for Large Item sets)

An efficient algorithm [4] called DELI is proposed for an incremental ARM. When database update occurs, DELI uses a sampling technique to decide whether it is necessary to generate a new set of association rules or not. If an estimate is small, it treats an old set of rules are a good approximation to the new set of rules. It waits until more changes are made to the database and DELI algorithm is applied again. If the estimate is large then the FUP2 algorithm is applied to generate a new set of rules. DELI finds to be more efficient than FUP2.

E. UWEP (Update with Early Pruning)

This algorithm [3] is another kind where incremental ARM takes place. However, it uses a property known as early pruning. It has an advantage over FUP-based algorithm as it is capable of pruning the supersets of originally mined frequent item set. When D is the dataset, the UWEP algorithm prunes as and when needed and does not wait until k-the iteration. This can improve its performance significantly. The rationale behind this is that early pruning can avoid unnecessary processing of certain records while focusing only on the incremental updates.

F. MAAP and PELICAN

MAAP [6] generates frequent itemsets of large size depending on the mined itemsets that are frequent. If an item set denoted as k-itemset is frequent, its subsets are also added to frequent item sets denote as L' according to the Apriori property. It results in the reduction of computational complexity. Afterward, other frequent itemsets are identified based on the concept of level-wise item set generation. The MAAP and PELICAN algorithms are closer to FUP2 but they aim at maintaining minimum frequent itemsets as the database is subjected to changes from time to time. MAAP computes maximum frequent item sets by using Apriori property while the PELICAN does the same using lattice decomposition and vertical database format. As these two makes use of maximum frequent item sets only, they are able to reduce space and time complexity while mining association rules incrementally.

G. IncA Algorithm

A new edition of Apriori algorithm for growing databases called Incremental Apriori (IncA)[12]. Incoming Transactions are being scanned and item sets are updated. The proposed algorithm saves running time and memory space.

H. IRARM (Incremental Relational Association Rule Mining) Algorithm

DRAR (Relational Association Rule mining Algorithm) [13]is proposed for mining association Rules. But the drawback is, it works only on static data base. Later IRARAM [14] is proposed for efficiently mining of all interesting Relational Association Rules from incremental data set .*IRARM* will use Relational Association Rules which are previously discovered from the original data set to compute the Relational Association Rules for the incremental data set.

IV. EXPERIMENTAL ANALYSIS AND DISCUSSION ON APRIORI-BASED INCREMENTAL MINING ALGORITHMS

The Performance of algorithms is tested using synthetic dataset and their behavior is recorded in a table and also shown graphically.

A. Comparing the performance of Apriori & FUP Algorithms

The performances of Apriori, FUP algorithms are tested using T1014D100 synthetic dataset. The following results were obtained, revealed in table 1 and shown in fig.3.

Table- I: Performance comparison of Apriori and FUP

Minimum support (%)	Execution time Ratio	
	Apriori	*FUP*
0.75	2.92	2.92
1.00	3.66	3.70
2.00	7.13	5.85
4.00	6.70	3.12
6.00	5.48	1.56

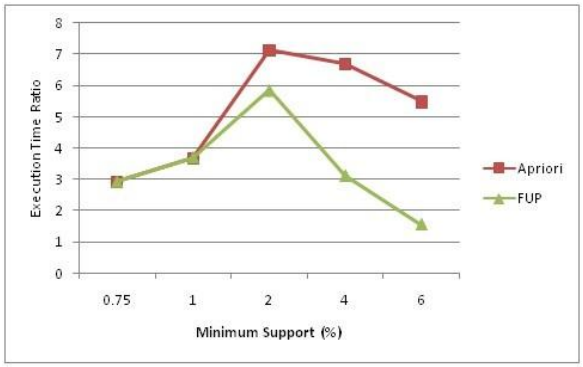

Fig.3. Performance comparison of Apriori and FUP

From the experimental result, we can conclude that FUP is faster than Apriori. For larger support, the execution time is very less for FUP when compared with Apriori.

B. Comparing the performance of Apriori & FUP2 Algorithms

The performances of Apriori, FUP &FUP2 algorithms are tested using T1014D100 synthetic dataset. Experimental results are revealed in Table II and shown in fig.4.

Table- II: Performance comparison of Apriori and FUP2

Support Threshold (%)	Execution time Ratio	
	Apriori	*FUP2*
1.00	896.44	283.12
1.25	692.53	200.00
1.50	563.29	153.25
1.75	489.06	127.27
2.00	442.35	127.27
2.25	419.22	80.52
2.50	392.14	72.73
2.75	369.01	75.32
3.00	361.60	70.13

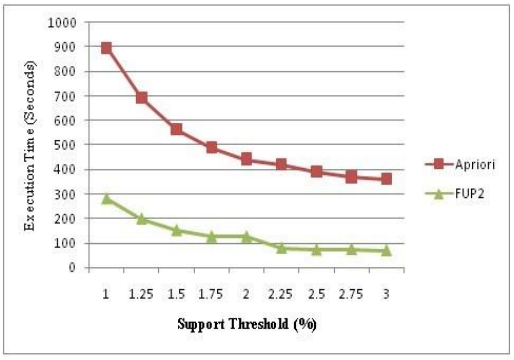

Fig.4. Performance comparison of Apriori and FUP2

From the Experimental results, it is found that the execution time for FUP2 is less when compared with Apriori for all the support values.

C. Comparing the performance of FUP, FUP2 & UWEP Algorithms

The performance of Apriori, FUP2 & UWEP algorithms is tested using synthetic dataset. Experimental results were revealed in Table III and shown in fig.5.

Table- III Performance Comparison of FUP, FUP2, and UWEP

Minimum support (%)	Execution time Ratio		
	FUP	*FUP2*	*UWEP*
1	1000	919.33	529.41
1.5	764.71	522.28	150.32
2	509.8	294.11	78.431
2.5	346.41	214.62	52.28
3	202.61	114.86	32.67

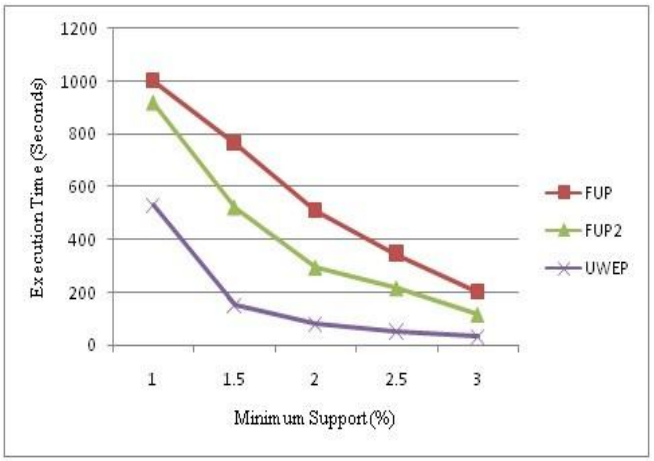

Fig.5. Performance comparison of FUP, FUP2 and UWEP algorithms

From the Experimental Results, we can see that Minimum support is varied between 1% and 6%. As support value is increased UWEP algorithm is far much better than FUP and FUP2.

D. Comparing the performance of Apriori, FUP2 & MAAP Algorithms

The performance of Apriori, FUP2 & MAAP algorithms is tested using T5.I2.D100k synthetic dataset. Experimental results were revealed in Table IV and shown in fig. 6.

Table- IV Performance Comparison of Apriori, FUP2, and MAAP

Minimum support (%)	Execution time Ratio		
	Apriori	*FUP2*	*MAAP*
1.4	13451.9	7807.2	4955.2
1.5	3716.0	2208.7	1608.6
1.6	2041.7	954.1	700.9
1.7	1268.1	596.0	436.6
1.8	856.8	408.6	302.8
1.9	625.7	312.8	235.6

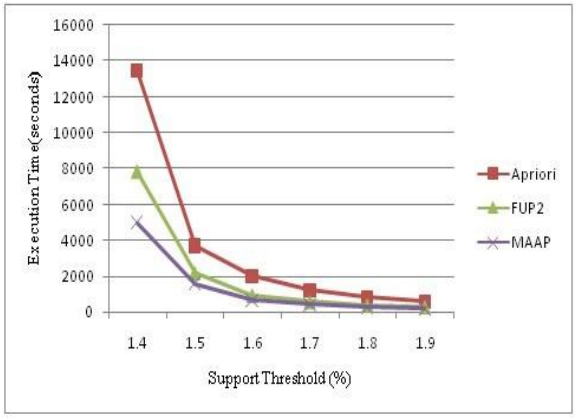

Fig .6.Performance comparison of Apriori, FUP2 and MAAP algorithms

From the experimental result, we can conclude that (i) Execution time of FUP2 and MAAP decrease, as the support values increases. (ii) The implementation time of MAAP algorithm is less than that of FUP2 and Apriori algorithm, for the same support (iii) The difference in execution times of MAAP algorithm and FUP2 reduces, as the support value increases.

E. Comparing the performance of Apriori, IncA Algorithms

To check the performance of IncA, Diana-Lucia Miholcaa coded the algorithm in Java and used MATLAB. To carry out tests a synthetic dataset CACM database containing 3204 transactions and around 2500 different items have been used in [12]. Experimental results were revealed in Table V and shown in fig. 7.

Table- V Performance Comparison of Apriori and IncA

Minimum support (%)	*Execution time in seconds*	
	Apriori	*IncA*
20	97200	75600
40	50400	21600
60	39600	14400

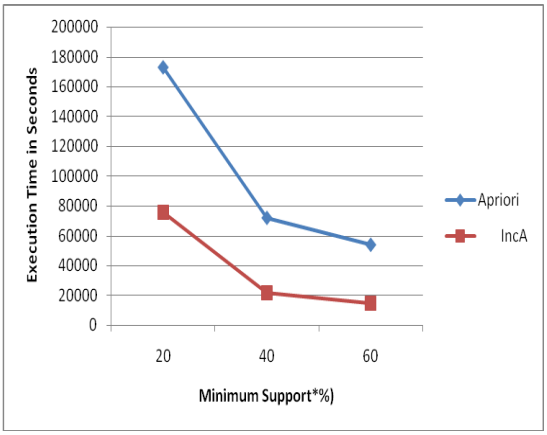

Fig .6.Performance comparison of Apriori, IncA algorithms

From the experimental result, we conclude that execution time for IncA is very less compared to Apriori algorithm.

F. Comparing the performance of DRAR and IRARM algorithms

The performance of DRAR and IRARM is compared using n a subset of Apis mellifera database from Eukaryotic Promoter Database in [15].The results obtained are shown in the table6 and in fig.7

Table- VI Performance Comparison of DRAR and IRARM

No. of Instances(n)	Execution time in seconds	
	DRAR	*IRARM*
1142	36.674	34.507
1714	36.492	27.416
2285	36.418	22.203
2856	36.492	17.904
3427	36.486	14.038
3998	36.448	10.509

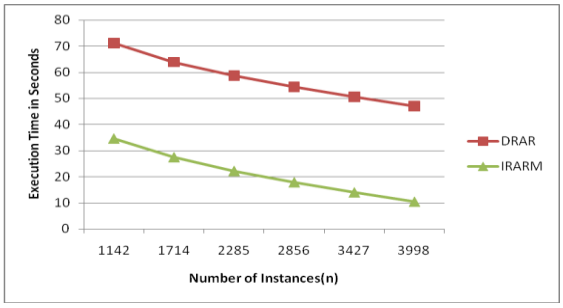

Fig .7.Performance comparison of DRAR, IRARM algorithms

From the experimental results, it is observed that, as the number of instances is increased, the execution time of IRARM is gradually decreased when compared to the execution time of DRAR.

V. PATTERN-GROWTH INCREMENTAL MINING ALGORITHMS

The general procedure for Pattern-Growth-based incremental ARM is illustrated in Fig.8 where the concept of dynamic ARM is witnessed.

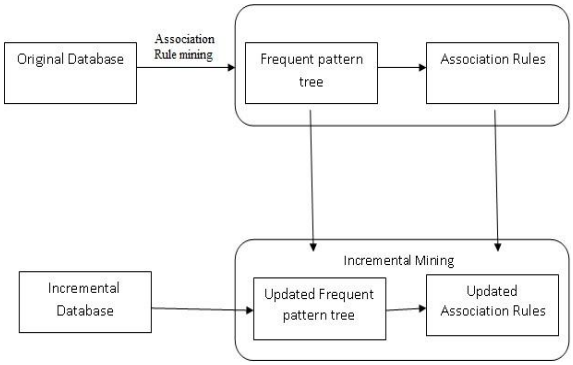

Fig.8. General Procedure for incremental ARM with Pattern-Growth approach

As shown in Fig.8, it is evident that first of all the original database is subjected to ARM. Afterwards, the incremental update made to database is subjected to incremental mining and the tree holding frequent patterns are updated and then the updated association rules are updated. Though it is a general procedure, it is followed by many Pattern-Growth incremental ARM algorithms [18],[19] but they use different structures that are described below.

A. DB-tree Algorithm

This algorithm [3] is another kind where incremental ARM takes place. However, it uses a property known as early pruning. It has an advantage over FUP-based algorithm as it is capable of pruning the supersets of originally mined frequent item set. When D is the dataset, the UWEP algorithm prunes as and when needed and does not wait until k-the iteration. This can improve its performance significantly. The rationale behind this is that early pruning can avoid unnecessary processing of certain records while focusing only on the incremental updates.

B. PotFP-tree Algorithm

PotFP-tree [7] follows different approach when compared to that of DB-tree algorithm. It stores only potentially frequent items apart from the frequent 1-itemsets. It uses the notation of tolerance and a new parameter ***t*** denoting tolerance to determine a potentially frequent item. As a matter of fact, FP-tree is a subset of PotFP-tree and also DB-tree for that matter. Therefore FP-tree is projected from either of them before extracting frequent item sets.

C. FELINE

CATS stand for Compressed and Arranged Transaction Sequence [8]. CATS tree has certain similarities with the features of FP-tree. Another important fact is that the DB-tree and the CATS tree are almost identical as they store all items irrespective of whether they are frequent. This seemingly significant feature makes the CATS tree right candidate to avoid re-scans of original database when incremental updates are needed. However, the way it is constructed differs from that of DB-tree and FP-tree. Ordering of global supports is employed in FP-tree while ordering local supports is employed by CATS-tree with respect to all frequent items in the path. From the original database CATS tree is built and it supports traversal in both directions in order to extract frequent item sets. But the construction of CATS tree is a complicated process and suitable for static databases in general.

D. CAN Tree (Canonical – Order Tree)

Originally CAN tree [9] is equipped with all tuples of given database. The items in the constructed tree are arranged in canonical order or alphabetical. In other words it supports an alphabetical order that has its utility in the data retrieval. Once the decision is made on the kind of ordering, that is followed by the tree while making subsequent changes in response to the changes made in the database. Then FP-growth kind of algorithm can be employed on the tree to generate frequent item sets. When database is subjected to frequent changes, CAN tree is an ideal candidate to reflect it. Neither it needs rescan of the original database fully nor CAN the reconstruction of a new tree in order to support incremental ARM. However, it needs more space in case of large CAN tree that also consumes more time for generating frequent item sets.

E. CP-tree (FP Growth algorithm)

It is the novel tree structure that is known as Compact Pattern Tree (CP-tree) [10]. It is convenient to capture data from database incrementally as database undergoes changes from time to time. Thus it achieves a structure known as frequency-descending structure. This tree is built in two steps. In the first step, inserts given transactions into the CP-tree which is according to the sort order that prevails with I-list. Then the frequency count is updated from time to time. The second step is for reconstruction. It is responsible to rearrange frequency-descending order of items besides updating the tree nodes to reflect new I-list. Alternative execution of these two phases makes the construction of CP tree and update of the same. CP-tree outperforms its predecessors in terms of memory consumption and execution time though construction process is complex and the cost of adjusting nodes in tree is high.

F. BIT_FP Growth algorithm

This algorithm is based on FP-tree.The BIT algorithm [11] performs merging of consecutive FP-trees in order to have final representative FP-tree that reflects the data present in the database. In case of large databases, the BIT algorithm is found to reduce execution time when it is compared with the performance of other incremental ARM algorithms. Its construction of tree takes less time though the merging process is complex and consumes more memory. Therefore, this algorithm is best used for batch processing.

G. FEEPAMT algorithm

An algorithm FEEPAMT [17] is proposed for mining incremental Association Rules and also reduces system complexity in terms of time and space.

To mine the association rules: For the given database, Canonical Ordered tree with Multiple Values Node tree or COMVAN [16] tree is constructed. Frequent item sets are generated from COMVAN tree.

VI. EXPERIMENTAL ANALYSIS AND DISCUSSION OF PATTERN-GROWTH INCREMENTAL MINING ALGORITHMS

The performance of algorithms, discussed in section 5 is analyzed using synthetic datasets. The results obtained are recorded in tables and also shown graphically.

A. Comparing the performance of FP-tree, DB-tree, and Pot-FP tree:

The performance of DB-tree, Pot-FPtree is compared with original FP-tree and the experimental results are recorded in a table 7 and graphical representation is shown in Fig.9. All these algorithms were tested using the dataset which is available at http://www.almaden.ibm.com/cs/quest/syndata.html.

Table- VII Performance Comparison of FP-tree, DB-tree, and Pot-Fp tree

Minimum support (%)	Execution time Ratio		
	FP-tree	*DB-tree*	*Pot-Fp tree*
0.1	54	44	42
0.2	51	44	40
0.3	49	44	38
0.5	43	43	34
0.7	39	43	30

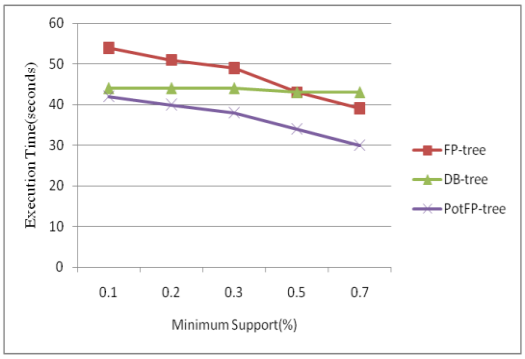

Fig. 9. Performance comparison of Pot-FP tree,DB-tree and FP-tree

From the experimental results, we observe that, implementation time decreases as the minimum support value is increased. For the minimum support value 0.5, the execution time of FP-tree is same that of DB-tree. The performance of PotFp-tree is superior compared to FP-tree and DB-tree.

F. Comparing the performance of FELINE and FP-Growth

The performance of FP-Growth and FELINE algorithm are evaluated using IBM data and obtained results are shown in table 8 and graphical representation is given in Fig.10.

Table- VIII Performance Comparison of FELINE and FP-Growth

Minimum support (%)	Execution time in seconds	
	FP-Growth	*FELINE*
0.00	158.87	142.62
0.10	42.55	26.31
0.20	18.44	5.04
0.30	18.44	0.57
0.40	18.44	0.29
0.50	15.60	0.00

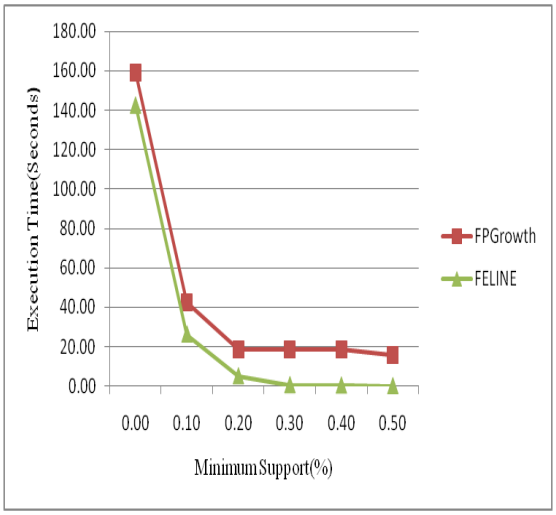

Fig.10.Performance comparison of FP-Growth and FELINE

From the experimental results, we can conclude that FELINE algorithm is better than FP-Growth for all the support values.

G. Comparing the performance of Can tree and CATS Tree

The performance of can tree and CATS tree is compared using IBM data. The following results were obtained shown in Table 9 and Fig. 11.

Table- IX Performance Comparison of Can tree and CATS Tree

Minimum support (%)	Execution time in seconds	
	Can tree	*CATS Tree*
5	149.15	249.14
10	26.05	39.56
15	8.35	13.75
20	4.16	12.27
25	2.72	8.11

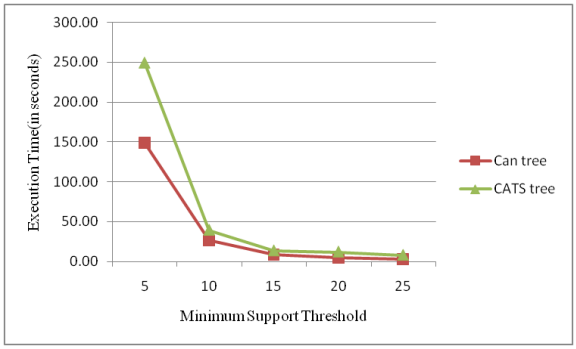

Fig.11 .Performance comparisons of Can tree and CATS Tree

From the experimental results, we can conclude that, for the higher support value execution time for can tree and CATS tree is same. Execution time for both the algorithms is highly varied for the lower support value. The performance of cantree is better when compared with CATS tree.

H. Comparing the performance of Can tree and BIT

The performance of can tree and BIT is compared using synthetic database. The following results were obtained shown in Table 10 and Figure 12.

Table- X Performance Comparison of Can tree and BIT

Data base size (in million transactions)	Execution time in seconds	
	Can tree	*BIT*
10	6.50	5.56
20	10.13	14.80
30	14.70	24.97
40	18.34	33.27
50	24.78	43.45
60	32.15	49.89
70	39.52	72.19
80	45.96	88.89
90	52.40	104.66
100	57.9028	118.562

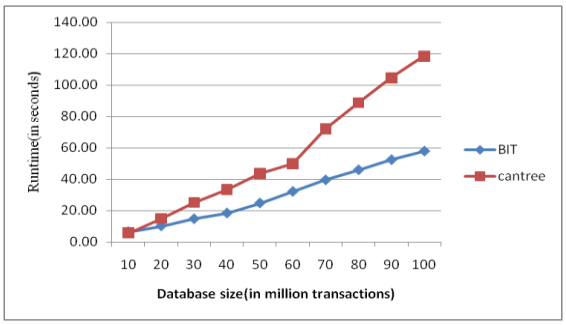

Fig.12.Performance comparison of Can tree and BIT

From the Experimental results, we can conclude that, performance of BIT and can tree is almost same for smaller size database. As the database size increases, there is a lot of difference in the performance of both the algorithms. Performance of BIT is better compared to the performance of can tree.

VII. CONCLUSION AND FUTURE SCOPE

In this paper, incremental ARM algorithms that are apriori-based and Patten-Growth are reviewed. Due to significant improvements in ARM process, the emergences of efficient algorithms that demonstrate automatic update in the generated association rules have emerged of late. It is the rationale that the work in this paper assumes significance. Many algorithms came into existence for incremental mining of rules. However, they are broadly categorized into Apriori-based and Pattern-Growth. A generic procedure for incremental ARM that employs apriori-based and Pattern-Growth approach is presented. It also provides recent developments in the academic thinking and inventions related to incremental association rule mining. In other words, this paper has provided many incremental ARM algorithms and their merits and demerits in terms of efficiency. The brief review of the incremental ARM algorithms can provide useful and quick insights on the present state-of-the-art. They can be used in real time applications in order to ensure that the rules are updated automatically and there is no need to rescan the entire database. Incremental database is sufficient to generate new rules and update the existing rules that have been generated. In future we intend to investigate the ability of these algorithms to deal with data that assumes qualities of big data.

REFERENCES

1. D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. Proceedings of the 12th International Conference on Data Engineering pages 106—114, February 1996.
2. D. Cheung, S. D. Lee, and B. Kao. A General Incremental Technique for Updating Discovered Association Rules. Proceedings of the Fifth International Conference On Database Systems for Advanced Applications pages 185—194, April 1997.
3. N. F. Ayan, A. U. Tansel, and M. E. Arkun. An Efficient Algorithm to Update Large Itemsets with Early Pruning. Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining pages 287—291, August 1999.
4. S.D.Lee, David W.Cheung and Ben Kao, Is Sampling Useful in Data Mining? A Case in the Maintenance of Discovered Association Rules, Data Mining and Knowledge Discovery, Volume 2 Issue 3, September 1998 Pages 233-262 Kluwer Academic Publishers Hingham, MA, USA.

5. S.Thomas, S. Bodagala, K.Alsabti and S. Ranka. An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. In KDD'97, New Port Beach, California, Aug.1997.
6. Z. Zhou and C. I. Ezeife. A Low-Scan Incremental Association Rule Maintenance Method based on the Apriori property. Proceedings of the 14th Canadian Conference on Artificial Intelligence, June 2001.
7. Ezeife C.I., Su Y. (2002), "Mining Incremental Association Rules with Generalized FP-Tree", In: Cohen R., Spencer B. (eds) Advances in Artificial Intelligence. AI 2002. Lecture Notes in Computer Science, vol 2338. Springer, Berlin, Heidelberg.
8. Cheung W, Zaiane OR (2003), "Incremental mining of frequent patterns without candidate generation or support constraint", In: Desai BC, Ng W (eds) Proceedings of the IDEAS 2003.IEEE Computer Society Press, Los Alamitos, CA, pp 111–116.
9. C. K. Leung, Q. I. Khan and T. Hoque. " CanTree: A Tree Structure for Efficient Incremental Mining of Frequent Patterns", Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), 2005.
10. Tanbeer S.K., Ahmed C.F., Jeong BS. Lee YK. (2008) , "CP-Tree: A Tree Structure for Single-Pass Frequent Pattern Mining", In: Washio T., Suzuki E., Ting K.M., Inokuchi A. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2008. Lecture Notes in Computer Science, vol 5012. Springer, Berlin, Heidelberg.
11. Totad, S.G., Geeta, R.B. & Prasad Reddy, P.V.G.D, "Batch incremental processing for FP-tree construction using FP-Growth algorithm", Knowledge and Information Systems, November 2012, Volume 33, Issue 2, pp 475–490 .https://doi.org/10.1007/s10115-012-0514-9.
12. Lydia Nahla, Driff & Drias, Habiba. (2017). An Efficient Incremental Mining Algorithm for Dynamic Databases. 1-12. 10.1007/978-3-319-58130-9_1.
13. S¸ erban, G., Campan, A., Czibula, I.G.. A programming interface for finding relational association rules. ˆ International Journal of Computers, Communications & Control 2006;I(S.):439–444.
14. Diana-Lucia Miholca, Gabriela Czibula, Liana Maria Crivei,A new incremental relational association rules mining approach, Procedia Computer Science,Volume 126,2018,Pages 126-135,ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.07.216.
15. Genomics, C.C... Eukaryothic promoter database. 2017.
16. Archana Gupta, Sanjeev Jain, Akhilesh Tiwari, "COMVAN: A Novel Data Structure for Storing Large Database for Incremental Association Mining", TECHNIA – International Journal of Computing Science and Communication Technologies, VOL.9 NO. 2, January. 2017 (ISSN 0974-3375), pp 1110- 1113.
17. Archana Gupta, Akhilesh Tiwari and Sanjeev Jain,Computer Science IJCSIS, Journal of. "A System for Incremental Association Rule Mining without Candidate Generation." IJCSIS July Vol 17 No 7, 2019.
18. Satyavathi N., Rama B., Nagaraju A.,"Research Travelogue-Performance Evaluation of Index support Item Set Mining Algorithms",International Journal of Pure and Applied Mathematics,Volume 120 No. 6 2018, 3641-3651,ISSN: 1314-3395 (on-line version),url: http://www.acadpubl.eu/hub/Special Issue.
19. Satyavathi N., Rama B., Nagaraju A. (2017) Incremental Updating of Mined Association Rules for Reflecting Record Insertions. In: Satapathy S., Prasad V., Rani B., Udgata S., Raju K. (eds) Proceedings of the First International Conference on Computational Intelligence and Informatics. Advances in Intelligent Systems and Computing, vol 507. Springer, Singapore.

AUTHORS PROILE

N.Satyavathi is Assistant professor at vaagdevi college of Engineering, obtained her Bachelors and Masters Degree in Computer Science and Engineering from JNTUH in 2005 and 2010 respectively. She is PhD research scholar in the Department of CSE at JNTUH.

Dr B.Rama has received MSc degree in Computer Science from Kakatiya University, Warangal. She obtained a PhD in Computer Science from Sri Padmavathi Mahila Viswa Vidhyalayam, Tirupati. Since 2010 she has been at the computer science department of Kakatiya University, where she serves as an Assistant professor. She published around 40 articles in various journals and reputed conferences like IEEE & Springer. She has patents published 5 in number for her credit. She authored Scholarly and research articles on a variety of areas related to Data mining, cloud computing, Machine learning and received a best paper award presented in International Conference.

Dr.A.Nagaraju is an Assistant Professor in the Department of Computer Science, Central University of Rajasthan. He received M.Sc (Pure Mathematics) from the Central University of Hyderabad, M.Tech in Computer Science and Technology from University of Mysore, and Ph.D. in Computer Science and Engineering from Osmania University, Hyderabad. His research area includes Wireless Sensor Networks, Ad-hoc Networks, Energy-Efficient Scheduling algorithms in Highly Distributed Systems, Artificial Intelligence and Machine Learning applications in Data Mining and Network Security. He has published 40 international conference papers, and 15 peer-reviewed journals indexed by SCIE, Scopus, and DBLP. He worked as Co-ordinator for the Department of Computer Science from Aug-2016 to Aug-2019, Central University of Rajasthan.